

## TRAITE DE OPERATION EN MATIERE BREVETS

PCT

NOTIFICATION DE LA RECEPTION DE  
L'EXEMPLAIRE ORIGINAL

(règle 24.2.a) du PCT)

Expéditeur: le BUREAU INTERNATIONAL

Destinataire:

CORLU, Bernard  
Bull S.A.  
PC58D20  
68, route de Versailles  
F-78434 Louveciennes Cedex  
FRANCE

Dir.  
Propri.

29 JAN. 2001

BULL S.A.

Date d'expédition (jour/mois/année) 17 janvier 2001 (17.01.01)	NOTIFICATION IMPORTANTE
Référence du dossier du déposant ou du mandataire PCT 3809/BC	Demande internationale no PCT/FR00/03193

Il est notifié au déposant que le Bureau international a reçu l'exemplaire original de la demande internationale précisée ci-après.

Nom(s) du ou des déposants et de l'Etat ou des Etats pour lesquels ils sont déposants:

BULL CP8 (pour tous les Etats désignés sauf US)  
GOIRE, Christian etc. (pour US seulement)

Date du dépôt international : 17 novembre 2000 (17.11.00)  
Date(s) de priorité revendiquée(s) : 17 novembre 1999 (17.11.99)  
Date de réception de l'exemplaire original  
par le Bureau international : 05 janvier 2001 (05.01.01)  
Liste des offices désignés :

EP : AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR  
National : BR,CA,CN,JP,US

## ATTENTION

Le déposant doit soigneusement vérifier les indications figurant dans la présente notification. En cas de divergence entre ces indications et celles que contient la demande internationale, il doit aviser immédiatement le Bureau international.

En outre, l'attention du déposant est appelée sur les renseignements donnés dans l'annexe en ce qui concerne

- ☒ les délais dans lesquels doit être abordée la phase nationale  
☒ la confirmation des désignations faites par mesure de précaution  
☐ les exigences relatives aux documents de priorité.

Une copie de la présente notification est envoyée à l'office récepteur et à l'administration chargée de la recherche internationale.

Bureau international de l'OMPI  
34, chemin des Colombettes  
1211 Genève 20, Suiss

n° de télécopieur (41-22) 740.14.35

Fonctionnaire autorisé

Kari Huynh-Khuong

n° de téléphone (41-22) 338.83.38

## RENSEIGNEMENTS CONCERNANT LES DELAIS DANS LESQUELS DOIT ETRE ABORDEE LA PHASE NATIONALE

Il est rappelé au déposant qu'il doit aborder la "phase nationale" auprès de chacun des offices désignés indiqués sur la notification de la réception de l'exemplaire original (formulaire PCT/IB/301) en payant les taxes nationales et en remettant les traductions, telles qu'elles sont prescrites par les législations nationales.

Le délai d'accomplissement de ces actes de procédure est de **20 MOIS** à compter de la date de priorité ou, pour les Etats désignés qui ont été élus par le déposant dans une demande d'examen préliminaire international ou dans une élection ultérieure, de **30 MOIS** à compter de la date de priorité, à condition que cette élection ait été effectuée avant l'expiration du 19<sup>e</sup> mois à compter de la date de priorité. Certains offices désignés (ou élus) ont fixé des délais qui expirent au-delà de 20 ou 30 mois à compter de la date de priorité. D'autres offices accordent une prolongation des délais ou un délai de grâce, dans certains cas moyennant le paiement d'une taxe supplémentaire.

En plus de ces actes de procédure, le déposant devra dans certains cas satisfaire à d'autres exigences particulières applicables dans certains offices. Il appartient au déposant de veiller à remplir en temps voulu les conditions requises pour l'ouverture de la phase nationale. La majorité des offices désignés n'envoient pas de rappel à l'approche de la date limite pour aborder la phase nationale.

Des informations détaillées concernant les actes de procédure à accomplir pour aborder la phase nationale auprès de chaque office désigné, les délais applicables et la possibilité d'obtenir une prolongation des délais ou un délai de grâce et toutes autres conditions applicables figurent dans le volume II du Guide du déposant du PCT. Les exigences concernant le dépôt d'une demande d'examen préliminaire international sont exposées dans le chapitre IX du volume I du Guide du déposant du PCT.

GR et ES sont devenues liées par le chapitre II du PCT le 7 septembre 1996 et le 6 septembre 1997, respectivement, et peuvent donc être élues dans une demande d'examen préliminaire international ou dans une élection ultérieure présentée le 7 septembre 1996 (ou à une date postérieure) ou le 6 septembre 1997 (ou à une date postérieure), respectivement, quelle que soit la date de dépôt de la demande internationale (voir le second paragraphe, ci-dessus).

Veuillez noter que seul un déposant qui est ressortissant d'un Etat contractant du PCT lié par le chapitre II ou qui y a son domicile peut présenter une demande d'examen préliminaire international.

## CONFIRMATION DES DESIGNATIONS FAITES PAR MESURE DE PRECAUTION

Seules les désignations expresses faites dans la requête conformément à la règle 4.9.a) figurent dans la présente notification. Il est important de vérifier si ces désignations ont été faites correctement. Des erreurs dans les désignations peuvent être corrigées lorsque des désignations ont été faites par mesure de précaution en vertu de la règle 4.9.b). Toute désignation ainsi faite peut être confirmée conformément aux dispositions de la règle 4.9.c) avant l'expiration d'un délai de 15 mois à compter de la date de priorité. En l'absence de confirmation, une désignation faite par mesure de précaution sera considérée comme retirée par le déposant. Il ne sera adressé aucun rappel ni invitation. Pour confirmer une désignation, il faut déposer une déclaration précisant l'Etat désigné concerné (avec l'indication de la forme de protection ou de traitement souhaitée) et payer les taxes de désignation et de confirmation. La confirmation doit parvenir à l'office récepteur dans le délai de 15 mois.

## EXIGENCES RELATIVES AUX DOCUMENTS DE PRIORITE

Pour les déposants qui n'ont pas encore satisfait aux exigences relatives aux documents de priorité, il est rappelé ce qui suit.

Lorsque la priorité d'une demande nationale, régionale ou internationale antérieure est revendiquée, le déposant doit présenter une copie de cette demande antérieure, certifiée conforme par l'administration auprès de laquelle elle a été déposée ("document de priorité"), à l'office récepteur (qui la transmettra au Bureau international) ou directement au Bureau international, avant l'expiration d'un délai de 16 mois à compter de la date de priorité, étant entendu que tout document de priorité peut être présenté au Bureau international avant la date de publication de la demande internationale, auquel cas ce document sera réputé avoir été reçu par le Bureau international le dernier jour du délai de 16 mois (règle 17.1.a)).

Lorsque le document de priorité est délivré par l'office récepteur, le déposant peut, au lieu de présenter ce document, demander à l'office récepteur de le préparer et de le transmettre au Bureau international. La requête à cet effet doit être formulée avant l'expiration du délai de 16 mois et peut être soumise au paiement d'une taxe (règle 17.1.b)).

Si le document de priorité en question n'est pas fourni au Bureau international, ou si la demande adressée à l'office récepteur de préparer et de transmettre le document de priorité n'a pas été faite (et la taxe correspondante acquittée, le cas échéant) avant l'expiration du délai applicable mentionné aux paragraphes précédents, tout Etat désigné peut ne pas tenir compte de la revendication de priorité; toutefois, aucun office désigné ne peut décider de ne pas tenir compte de la revendication de priorité avant d'avoir donné au déposant la possibilité de remettre le document de priorité dans un délai raisonnable en l'espèce.

Lorsque plusieurs priorités sont revendiquées, la date de priorité à prendre en considération aux fins du calcul du délai de 16 mois est la date du dépôt de la demande la plus ancienne dont la priorité est revendiquée.

## TRAITE DE OPERATION EN MATIERE BREVETS

PCT

NOTIFICATION RELATIVE  
A LA PRESENTATION OU A LA TRANSMISSION  
DU DOCUMENT DE PRIORITE

(instruction administrative 411 du PCT)

Expéditeur : le BUREAU INTERNATIONAL

Destinataire:

CORLU, Bernard  
Bull S.A.  
PC58D20  
68, route de Versailles  
F-78434 Louveciennes Cedex  
FRANCE

Date d'expédition (jour/mois/année) 17 janvier 2001 (17.01.01)	NOTIFICATION IMPORTANTE
Référence du dossier du déposant ou du mandataire PCT 3809/BC	
Demande internationale no PCT/FR00/03193	Date du dépôt international (jour/mois/année) 17 novembre 2000 (17.11.00)
Date de publication internationale (jour/mois/année) Pas encore publiée	Date de priorité (jour/mois/année) 17 novembre 1999 (17.11.99)
Déposant BULL CP8 etc	

- La date de réception (sauf lorsque les lettres "NR" figurent dans la colonne de droite) par le Bureau international du ou des documents de priorité correspondant à la ou aux demandes énumérées ci-après est notifiée au déposant. Sauf indication contraire consistant en un astérisque figurant à côté d'une date de réception, ou les lettres "NR", dans la colonne de droite, le document de priorité en question a été présenté ou transmis au Bureau international d'une manière conforme à la règle 17.1.a) ou b).
- Ce formulaire met à jour et remplace toute notification relative à la présentation ou à la transmission du document de priorité qui a été envoyée précédemment.
- Un astérisque(\*) figurant à côté d'une date de réception dans la colonne de droite signale un document de priorité présenté ou transmis au Bureau international mais de manière non conforme à la règle 17.1.a) ou b). Dans ce cas, l'attention du déposant est appelée sur la règle 17.1.c) qui stipule qu'aucun office désigné ne peut décider de ne pas tenir compte de la revendication de priorité avant d'avoir donné au déposant la possibilité de remettre le document de priorité dans un délai raisonnable en l'espèce.
- Les lettres "NR" figurant dans la colonne de droite signalent un document de priorité que le Bureau international n'a pas reçu ou que le déposant n'a pas demandé à l'office récepteur de préparer et de transmettre au Bureau international, conformément à la règle 17.1.a) ou b), respectivement. Dans ce cas, l'attention du déposant est appelée sur la règle 17.1.c) qui stipule qu'aucun office désigné ne peut décider de ne pas tenir compte de la revendication de priorité avant d'avoir donné au déposant la possibilité de remettre le document de priorité dans un délai raisonnable en l'espèce.

<u>Date de priorité</u>	<u>Demande de priorité n°</u>	<u>Pays, office régional ou office récepteur selon le PCT</u>	<u>Date de réception du document de priorité</u>
17 nove 1999 (17.11.99)	99/14454	FR	05 janv 2001 (05.01.01)

Bureau international de l'OMPI  
34, chemin des Colombettes  
1211 Genève 20, Suisse

no de télécopieur (41-22) 740.14.35

Fonctionnaire autorisé:

Kari Huynh-Khuong

no de téléphone (41-22) 338.83.38

003776073

Direction de la  
Propriété Intellectuelle

# TRAITE DE OPERATION EN MATIERE BREVETS

5 JUIN 2001

BULL S.A.

PCT

## AVIS INFORMANT LE DEPOSANT DE LA COMMUNICATION DE LA DEMANDE INTERNATIONALE AUX OFFICES DESIGNES

(règle 47.1.c), première phrase, du PCT)

Expéditeur: le BUREAU INTERNATIONAL

Destinataire:

CORLU, Bernard  
Bull S.A.  
PC58D20  
68, route de Versailles  
F-78434 Louveciennes Cedex  
FRANCE

Date d'expédition (jour/mois/année) 25 mai 2001 (25.05.01)		AVIS IMPORTANT	
Référence du dossier du déposant ou du mandataire PCT 3809/BC			
Demande internationale no PCT/FR00/03193	Date du dépôt international (jour/mois/année) 17 novembre 2000 (17.11.00)	Date de priorité (jour/mois/année) 17 novembre 1999 (17.11.99)	
Déposant BULL CP8 etc			

1. Il est notifié par la présente qu'à la date indiquée ci-dessus comme date d'expédition de cet avis, le Bureau international a communiqué, comme le prévoit l'article 20, la demande internationale aux offices désignés suivants:  
US

Conformément à la règle 47.1.c), troisième phrase, ces offices acceptent le présent avis comme preuve déterminante du fait que la communication de la demande internationale a bien eu lieu à la date d'expédition indiquée plus haut, et le déposant n'est pas tenu de remettre de copie de la demande internationale à l'office ou aux offices désignés.

2. Les offices désignés suivants ont renoncé à l'exigence selon laquelle cette communication doit être effectuée à cette date:  
BR,CA,CN,EP,JP

La communication sera effectuée seulement sur demande de ces offices. De plus, le déposant n'est pas tenu de remettre de copie de la demande internationale aux offices en question (règle 49.1)a-bis)).

3. Le présent avis est accompagné d'une copie de la demande internationale publiée par le Bureau international le 25 mai 2001 (25.05.01) sous le numéro WO 01/37085

### RAPPEL CONCERNANT LE CHAPITRE II (article 31.2)a) et règle 54.2)

Si le déposant souhaite reporter l'ouverture de la phase nationale jusqu'à 30 mois (ou plus pour ce qui concerne certains offices) à compter de la date de priorité, la demande d'examen préliminaire international doit être présentée à l'administration compétente chargée de l'examen préliminaire international avant l'expiration d'un délai de 19 mois à compter de la date de priorité.

Il appartient exclusivement au déposant de veiller au respect du délai de 19 mois.

Il est à noter que seul un déposant qui est ressortissant d'un Etat contractant du PCT lié par le chapitre II ou qui y a son domicile peut présenter une demande d'examen préliminaire international.

### RAPPEL CONCERNANT L'OUVERTURE DE LA PHASE NATIONALE (article 22 ou 39.1))

Si le déposant souhaite que la demande internationale procède en phase nationale, il doit, dans le délai de 20 mois ou de 30 mois, ou plus pour ce qui concerne certains offices, accomplir les actes mentionnés dans ces dispositions auprès de chaque office désigné ou élu.

Pour d'autres informations importantes concernant les délais et les actes à accomplir pour l'ouverture de la phase nationale, voir l'annexe du formulaire PCT/IB/301 (Notification de la réception de l'exemplaire original) et le volume II du Guide du déposant du PCT.

<p>Bureau international de l'OMPI 34, chemin des Colmbettes 1211 Genève 20, Suisse</p> <p>no de télécopieur (41-22) 740.14.35</p>	<p>Fonctionnaire autorisé J. Zahra</p> <p>no de téléphone (41-22) 338.83.38</p>
---	---

# PCT

## REQUÊTE

Le soussigné requiert que la présente demande internationale soit traitée conformément au Traité de coopération en matière de brevets.

Réservé à l'office récepteur

Demande internationale n°

Date du dépôt international

Nom de l'office récepteur et "Demande internationale PCT"

Référence du dossier du déposant ou du mandataire (facultatif)  
(12 caractères au maximum) **PCT 3809/BC**

**Cadre n° I TITRE DE L'INVENTION** Procédé de chargement d'applications dans un système embarqué multi-application muni de ressources de traitement de données, système embarqué correspondant, et procédé d'exécution d'une application d'un système embarqué correspondant.

### Cadre n° II DÉPOSANT

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'État où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

**BULL CP8  
68, route de Versailles  
BP 45  
78430 LOUVECIENNES  
FRANCE**

☐ Cette personne est aussi inventeur.

n° de téléphone

**(33) 1 39.66.61.76**

n° de télécopieur

**(33) 1 39.66.61.73**

n° de téléimprimeur

Nationalité (nom de l'État) :

**FRANCE**

Domicile (nom de l'État) :

**FRANCE**

Cette personne est déposant pour :

☐ tous les États désignés

☒ tous les États désignés sauf les États-Unis d'Amérique

☐ les États-Unis d'Amérique seulement

☐ les États indiqués dans le cadre supplémentaire

### Cadre n° III AUTRE(S) DÉPOSANT(S) OU (AUTRE(S)) INVENTEUR(S)

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'État où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

**Goire Christian  
8 allée du Mail  
78340 LES CLAYES SOUS BOIS  
FRANCE**

Cette personne est :

☐ déposant seulement

☒ déposant et inventeur

☐ inventeur seulement  
(Si cette case est cochée, ne pas remplir la suite.)

Nationalité (nom de l'État) :

**FRANCE**

Domicile (nom de l'État) :

**FRANCE**

Cette personne est déposant pour :

☐ tous les États désignés

☐ tous les États désignés sauf les États-Unis d'Amérique

☒ les États-Unis d'Amérique seulement

☐ les États indiqués dans le cadre supplémentaire

☐ D'autres déposants ou inventeurs sont indiqués sur une feuille annexe.

### Cadre n° IV MANDATAIRE OU REPRÉSENTANT COMMUN; OU ADRESSE POUR LA CORRESPONDANCE

La personne dont l'identité est donnée ci-dessous est/à été désignée pour agir au nom du ou des déposants auprès des autorités internationales compétentes, comme :

☒ mandataire

☐ représentant commun

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays.)

**BULL S.A  
CORLU Bernard  
PC58D20 / 68, route de Versailles  
F- 78434 LOUVECIENNES Cedex (FRANCE)**

n° de téléphone

**(33) 1 39.66.61.76**

n° de télécopieur

**(33) 1 39.66.61.73**

n° de téléimprimeur

☐ Adresse pour la correspondance : cocher cette case lorsque aucun mandataire ni représentant commun n'est/n'a été désigné et que l'espace ci-dessus est utilisé pour indiquer une adresse spéciale à laquelle la correspondance doit être envoyée.

## Suite du cadre n° III AUTRE(S) DÉPOSANT(S) OU (AUTRE(S)) INVENTEUR(S)

*Si aucun des sous-cadres suivants n'est utilisé, cette feuille ne doit pas être incluse dans la requête.*

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'État où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

Billon Jean-Paul  
96 Uranus terrace  
SAN FRANCISCO, CA 94114  
USA

Cette personne est :

- ☐ déposant seulement  
☒ déposant et inventeur  
☐ inventeur seulement  
(Si cette case est cochée, ne pas remplir la suite.)

Nationalité (nom de l'État) :

FRANCE

Domicile (nom de l'État) :

USA

Cette personne est déposant pour :

- ☐ tous les États désignés ☐ tous les États désignés sauf les États-Unis d'Amérique ☒ les États-Unis d'Amérique seulement ☐ les États indiqués dans le cadre supplémentaire

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'État où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

Cette personne est :

- ☐ déposant seulement  
☐ déposant et inventeur  
☐ inventeur seulement  
(Si cette case est cochée, ne pas remplir la suite.)

Nationalité (nom de l'État) :

Domicile (nom de l'État) :

Cette personne est déposant pour :

- ☐ tous les États désignés ☐ tous les États désignés sauf les États-Unis d'Amérique ☐ les États-Unis d'Amérique seulement ☐ les États indiqués dans le cadre supplémentaire

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'État où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

Cette personne est :

- ☐ déposant seulement  
☐ déposant et inventeur  
☐ inventeur seulement  
(Si cette case est cochée, ne pas remplir la suite.)

Nationalité (nom de l'État) :

Domicile (nom de l'État) :

Cette personne est déposant pour :

- ☐ tous les États désignés ☐ tous les États désignés sauf les États-Unis d'Amérique ☐ les États-Unis d'Amérique seulement ☐ les États indiqués dans le cadre supplémentaire

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'État où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

Cette personne est :

- ☐ déposant seulement  
☐ déposant et inventeur  
☐ inventeur seulement  
(Si cette case est cochée, ne pas remplir la suite.)

Nationalité (nom de l'État) :

Domicile (nom de l'État) :

Cette personne est déposant pour :

- ☐ tous les États désignés ☐ tous les États désignés sauf les États-Unis d'Amérique ☐ les États-Unis d'Amérique seulement ☐ les États indiqués dans le cadre supplémentaire

☐ D'autres déposants ou inventeurs sont indiqués sur une autre feuille annexe.

**Cadre n° V DÉSIGNATION D'ÉTATS**

Les désignations suivantes sont faites conformément à la règle 4.9.a) (cocher les cases appropriées au moins doit l'être) :

**Brevet régional**

- ☐ AP Brevet ARIPO : GH Ghana, GM Gambie, KE Kenya, LS Lesotho, MW Malawi, SD Soudan, SL Sierra Leone, SZ Swaziland, TZ République-Unie de Tanzanie, UG Ouganda, ZW Zimbabwe et tout autre État qui est un État contractant du Protocole de Harare et du PCT
- ☐ EA Brevet eurasiatique : AM Arménie, AZ Azerbaïdjan, BY Bélarus, KG Kirghizistan, KZ Kazakhstan, MD République de Moldova, RU Fédération de Russie, TJ Tadjikistan, TM Turkménistan et tout autre État qui est un État contractant de la Convention sur le brevet eurasiatique et du PCT
- ☒ EP Brevet européen : AT Autriche, BE Belgique, CH et LI Suisse et Liechtenstein, CY Chypre, DE Allemagne, DK Danemark, ES Espagne, FI Finlande, FR France, GB Royaume-Uni, GR Grèce, IE Irlande, IT Italie, LU Luxembourg, MC Monaco, NL Pays-Bas, PT Portugal, SE Suède et tout autre État qui est un État contractant de la Convention sur le brevet européen et du PCT
- ☐ OA Brevet OAPI : BF Burkina Faso, BJ Bénin, CF République centrafricaine, CG Congo, CI Côte d'Ivoire, CM Cameroun, GA Gabon, GN Guinée, GW Guinée-Bissau, ML Mali, MR Mauritanie, NE Niger, SN Sénégal, TD Tchad, TG Togo et tout autre État qui est un État membre de l'OAPI et un État contractant du PCT (si une autre forme de protection ou de traitement est souhaitée, le préciser sur la ligne pointillée) . . . . .

**Brevet national (si une autre forme de protection ou de traitement est souhaitée, le préciser sur la ligne pointillée) :**

- |  |   |
|--|---|
| <input type="checkbox"/> AE Émirats arabes unis                        | <input type="checkbox"/> LR Liberia                               |
| <input type="checkbox"/> AL Albanie                                    | <input type="checkbox"/> LS Lesotho                               |
| <input type="checkbox"/> AM Arménie                                    | <input type="checkbox"/> LT Lituanie                              |
| <input type="checkbox"/> AT Autriche                                   | <input type="checkbox"/> LU Luxembourg                            |
| <input type="checkbox"/> AU Australie                                  | <input type="checkbox"/> LV Lettonie                              |
| <input type="checkbox"/> AZ Azerbaïdjan                                | <input type="checkbox"/> MA Maroc                                 |
| <input type="checkbox"/> BA Bosnie-Herzégovine                         | <input type="checkbox"/> MD République de Moldova                 |
| <input type="checkbox"/> BB Barbade                                    | <input type="checkbox"/> MG Madagascar                            |
| <input type="checkbox"/> BG Bulgarie                                   | <input type="checkbox"/> MK Ex-République yougoslave de Macédoine |
| <input checked="" type="checkbox"/> BR Brésil                          | <input type="checkbox"/> MN Mongolie                              |
| <input type="checkbox"/> BY Bélarus                                    | <input type="checkbox"/> MW Malawi                                |
| <input checked="" type="checkbox"/> CA Canada                          | <input type="checkbox"/> MX Mexique                               |
| <input type="checkbox"/> CH et LI Suisse et Liechtenstein              | <input type="checkbox"/> NO Norvège                               |
| <input checked="" type="checkbox"/> CN Chine                           | <input type="checkbox"/> NZ Nouvelle-Zélande                      |
| <input type="checkbox"/> CR Costa Rica                                 | <input type="checkbox"/> PL Pologne                               |
| <input type="checkbox"/> CU Cuba                                       | <input type="checkbox"/> PT Portugal                              |
| <input type="checkbox"/> CZ République tchèque                         | <input type="checkbox"/> RO Roumanie                              |
| <input type="checkbox"/> DE Allemagne                                  | <input type="checkbox"/> RU Fédération de Russie                  |
| <input type="checkbox"/> DK Danemark                                   | <input type="checkbox"/> SD Soudan                                |
| <input type="checkbox"/> DM Dominique                                  | <input type="checkbox"/> SE Suède                                 |
| <input type="checkbox"/> EE Estonie                                    | <input type="checkbox"/> SG Singapour                             |
| <input type="checkbox"/> ES Espagne                                    | <input type="checkbox"/> SI Slovénie                              |
| <input type="checkbox"/> FI Finlande                                   | <input type="checkbox"/> SK Slovaquie                             |
| <input type="checkbox"/> GB Royaume-Uni                                | <input type="checkbox"/> SL Sierra Leone                          |
| <input type="checkbox"/> GD Grenade                                    | <input type="checkbox"/> TJ Tadjikistan                           |
| <input type="checkbox"/> GE Géorgie                                    | <input type="checkbox"/> TM Turkménistan                          |
| <input type="checkbox"/> GH Ghana                                      | <input type="checkbox"/> TR Turquie                               |
| <input type="checkbox"/> GM Gambie                                     | <input type="checkbox"/> TT Trinité-et-Tobago                     |
| <input type="checkbox"/> HR Croatie                                    | <input type="checkbox"/> TZ République-Unie de Tanzanie           |
| <input type="checkbox"/> HU Hongrie                                    | <input type="checkbox"/> UA Ukraine                               |
| <input type="checkbox"/> ID Indonésie                                  | <input type="checkbox"/> UG Ouganda                               |
| <input type="checkbox"/> IL Israël                                     | <input checked="" type="checkbox"/> US États-Unis d'Amérique      |
| <input type="checkbox"/> IN Inde                                       | <input type="checkbox"/> UZ Ouzbékistan                           |
| <input type="checkbox"/> IS Islande                                    | <input type="checkbox"/> VN Viet Nam                              |
| <input checked="" type="checkbox"/> JP Japon                           | <input type="checkbox"/> YU Yougoslavie                           |
| <input type="checkbox"/> KE Kenya                                      | <input type="checkbox"/> ZA Afrique du Sud                        |
| <input type="checkbox"/> KG Kirghizistan                               | <input type="checkbox"/> ZW Zimbabwe                              |
| <input type="checkbox"/> KP République populaire démocratique de Corée |   |
| <input type="checkbox"/> KR République de Corée                        |   |
| <input type="checkbox"/> KZ Kazakhstan                                 |   |
| <input type="checkbox"/> LC Sainte-Lucie                               |   |
| <input type="checkbox"/> LK Sri Lanka                                  |   |

Cases réservées pour la désignation d'États qui sont devenus parties au PCT après la publication de la présente feuille :

**Déclaration concernant les désignations de précaution :** outre les désignations faites ci-dessus, le déposant fait aussi conformément à la règle 4.9.b) toutes les désignations qui seraient autorisées en vertu du PCT, à l'exception de toute désignation indiquée dans le cadre supplémentaire comme étant exclue de la portée de cette déclaration. Le déposant déclare que ces désignations additionnelles sont faites sous réserve de confirmation et que toute désignation qui n'est pas confirmée avant l'expiration d'un délai de 15 mois à compter de la date de priorité doit être considérée comme retirée par le déposant à l'expiration de ce délai. (La confirmation (y compris les taxes) doit parvenir à l'office récepteur dans le délai de 15 mois.)

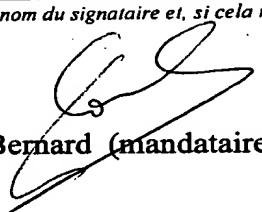
Cadre n° VI REVENDEICATION DE <span style="background-color: black; color: black;">PRIÉTÉ</span>		<input type="checkbox"/> D'... revendications de priorité sont indiquées dans le cadre supplémentaire.		
Date de dépôt de la demande antérieure (jour/mois/année)	Numéro de la demande antérieure	Lorsque la demande antérieure est une :		
		demande nationale : pays	demande régionale :* office régional	demande internationale : office récepteur
(1) 17 novembre 1999 (17.11.1999)	99 14454	FRANCE		
(2)				
(3)				

☒ L'office récepteur est prié de préparer et de transmettre au Bureau international une copie certifiée conforme de la ou des demandes antérieures (seulement si la demande antérieure a été déposée auprès de l'office qui, aux fins de la présente demande internationale, est l'office récepteur) indiquées ci-dessus au(x) point(s) : 1

\* Si la demande antérieure est une demande ARIPO, il est obligatoire d'indiquer dans le cadre supplémentaire au moins un pays partie à la Convention de Paris pour la protection de la propriété industrielle pour lequel cette demande antérieure a été déposée (règle 4.10.b)ii). Voir le cadre supplémentaire.

Cadre n° VII ADMINISTRATION CHARGÉE DE LA RECHERCHE INTERNATIONALE			
Choix de l'administration chargée de la recherche internationale (ISA) (si plusieurs administrations chargées de la recherche internationale sont compétentes pour procéder à la recherche internationale, indiquer l'administration choisie; le code à deux lettres peut être utilisé) : ISA /	Demande d'utilisation des résultats d'une recherche antérieure; mention de cette recherche (si une recherche antérieure a été effectuée par l'administration chargée de la recherche internationale ou demandée à cette dernière) : <div style="display: flex; justify-content: space-between;"> <span>Date (jour/mois/année) 17.11.99</span> <span>Numéro 99 14454 FA</span> <span>Pays (ou office régional) FR non publié</span> </div>		

Cadre n° VIII BORDEREAU; LANGUE DE DÉPÔT	
La présente demande internationale contient le nombre de feuilles suivant :  requête : 04 description (sauf partie réservée au listage des séquences) : 25 revendications : 05 abrégé : 01 dessins : 04 partie de la description réservée au listage des séquences : _____  Nombre total de feuilles : 39	Le ou les éléments cochés ci-après sont joints à la présente demande internationale : 1. <input type="checkbox"/> feuille de calcul des taxes 2. <input checked="" type="checkbox"/> pouvoir distinct signé 3. <input type="checkbox"/> copie du-pouvoir général; numéro de référence, le cas échéant : 4. <input type="checkbox"/> explication de l'absence d'une signature 5. <input checked="" type="checkbox"/> document(s) de priorité indiqué(s) dans le cadre n° VI au(x) point(s) 1 6. <input type="checkbox"/> traduction de la demande internationale en (langue) : 7. <input type="checkbox"/> indications séparées concernant des micro-organismes ou autre matériel biologique déposés 8. <input type="checkbox"/> listage des séquences de nucléotides ou d'acides aminés sous forme déchiffrable par ordinateur 9. <input type="checkbox"/> autres éléments (préciser) : <b>Rapport de Recherche NON PUBLIE</b>
Figure des dessins qui doit accompagner l'abrégé : -	Langue de dépôt de la demande internationale : FRANCAIS

Cadre n° IX SIGNATURE DU DÉPOSANT OU DU MANDATAIRE	
À côté de chaque signature, indiquer le nom du signataire et, si cela n'apparaît pas clairement à la lecture de la requête, à quel titre l'intéressé signe.   <b>CORLU Bernard (mandataire)</b>	

Réservé à l'office récepteur	
1. Date effective de réception des pièces supposées constituer la demande internationale :  3. Date effective de réception, rectifiée en raison de la réception ultérieure, mais dans les délais, de documents ou de dessins complétant ce qui est supposé constituer la demande internationale :  4. Date de réception, dans les délais, des corrections demandées selon l'article 11.2) du PCT :	2. Dessins : <input type="checkbox"/> reçus :  <input type="checkbox"/> non reçus :
5. Administration chargée de la recherche internationale (si plusieurs sont compétentes) : ISA /	6. <input type="checkbox"/> Transmission de la copie de recherche différée jusqu'au paiement de la taxe de recherche.

Réservé au Bureau international	
Date de réception de l'exemplaire original par le Bureau international :	



**Procédé de chargement d'applications dans un système embarqué multi-application muni de ressources de traitement de données, système embarqué correspondant, et procédé d'exécution d'une application d'un système embarqué correspondant**

La présente invention concerne un procédé de chargement d'applications dans un système embarqué multi-application muni de ressources de traitement de données, le système embarqué correspondant, et le procédé d'exécution d'une application d'un système embarqué correspondant.

La présente invention concerne plus particulièrement la réalisation de  
pare-feu entre modules partageant le même espace mémoire dans des  
systèmes embarqués sur des objets portables multi-application utilisant un  
pseudocode intermédiaire et une machine virtuelle associée.

La technologie "Java" (marque déposée) introduite par la société "Sun" est basée sur un langage de programmation orienté objet "Java" et une machine virtuelle associée. Cette technologie a été développée sur des stations ou PC (Personal Computer), appelées ci-après plate-forme "Java" classique, possédant une puissance CPU et une mémoire importante de l'ordre du méga byte ou mégaoctet.

20 Depuis quelques années, les concepts de la technologie "Java" ont  
été repris et adaptés pour fonctionner sur des systèmes embarqués dans  
des objets portables, par exemple au format de carte de crédit ou  
micromodule SIM incorporant un microprocesseur et appelée communément  
carte à puce, ou encore de téléphones portables GSM (Global System for  
25 Mobile Communication), cartes PCMCIA ou tous autres terminaux portables.  
Par la suite nous utiliserons le terme carte pour désigner l'un quelconque de  
ces objets portables. La programmation des systèmes embarqués, jusqu'ici  
réalisée en assembleur, est désormais possible dans un langage évolué  
comme "Java", et permet de faciliter et d'accélérer le développement  
30 d'applications clientes.

Ce nouveau type de plate-forme spécifique aux systèmes embarqués d'un objet portable, appelé ci-après plate-forme spécifique, constitue un sous-ensemble de la plate-forme classique. Les différences résultent du fait de l'environnement réduit des systèmes embarqués. De manière classique, tel que représenté à la figure 4a, une carte à puce (10) comprend un système d'entrée et de sortie (11) relié au microprocesseur (14), une mémoire volatile RAM (12) (Random Access Memory) une mémoire non volatile constituée par une mémoire morte ROM (13) (Read Only Memory) et une mémoire non volatile programmable (15) constituée d'une flash RAM ou EEPROM (Electrically Erasable Programmable Read Only Memory). L'ensemble de ces éléments est relié au microprocesseur par un BUS de liaison. A titre d'exemple, une carte à puce comprend dans le meilleurs des cas, en utilisant les nouveaux composants actuellement existants, une mémoire ROM, une mémoire EEPROM de 32 kilooctets ou kilo bytes, et une mémoire RAM de 2 Kilo Bytes.

Dans un système "Java" classique, une application est écrite sur une station ou PC. Son code source est compilé dans un pseudocode intermédiaire, appelé "Java Byte Code" ou byte code, qui est indépendant du code machine de la plate-forme utilisée. L'application est ensuite téléchargée sur la plate-forme cible ou plate-forme "Java" classique. L'application chargée, constituée d'un ensemble de classes dites classes clients dont les méthodes ont été compilées dans le pseudocode, s'appuie sur les interfaces de programmation applicatives ou Interface pour la Programmation d'Application ou API (Application Programming Interface). Les interfaces de programmation applicatives permettent d'uniformiser l'interface utilisateur, de contrôler un éditeur, un clavier ou une imprimante par exemple. La machine virtuelle d'une plate-forme classique comprend un vérifieur de pseudocode, un chargeur dynamique de classe, un interpréteur de pseudocode qui permet la traduction en code machine et un gestionnaire de sécurité.

La principale différence entre une machine virtuelle classique et une machine virtuelle d'un système embarqué, appelée machine virtuelle

spécifique, est due au fait que la machine virtuelle d'un système embarqué est décomposée en deux parties séparées. Suivant la figure 4b, la machine virtuelle comprend une partie (30) hors de la plate-forme spécifique (40), appelée machine virtuelle hors plate-forme ("off-platform"), comprenant un convertisseur (32), et une partie (41) dans la carte constituant la plate-forme spécifique (40), appelée machine virtuelle embarquée (41) ("on-platform") incluant l'interpréteur de pseudocode.

Ainsi, dans le cas d'une plate-forme spécifique, le programme source (21) de l'application est écrit, compilé en pseudocode intermédiaire par un compilateur (22), et vérifié par un vérifieur (31) de pseudocode intermédiaire sur une station (20) classique, puis converti par le convertisseur (32), placé sur la même station (20) ou une autre station. Après un éventuel passage par un signeur (34), l'application est ensuite téléchargée sur la mémoire volatile programmable électriquement et éventuellement effaçable électriquement EEPROM (15) de l'objet portable ou plate-forme spécifique (40). Ce chargement est effectué par un chargeur comportant une partie hors plate-forme appelée téléchargeur (33) et une partie sur la plate-forme spécifique appelée chargeur (42). Contrairement à ce qui existe sur une plate-forme classique pour une station, la machine virtuelle (41) d'une plate-forme spécifique (40), placée en mémoire ROM avec le système d'exploitation (48) (Operating System) ne peut comporter de vérifieur de pseudocode intermédiaire, celui-ci étant trop lourd pour être stocké et/ou exécuté dans l'objet portable. La plate-forme spécifique (40) ne contient pas non plus de chargeur dynamique de classes. En effet, pour le domaine d'application de l'invention, sur la machine virtuelle (30) hors plate-forme, le vérifieur (31) vérifie que les classes compilées sont bien formées et vérifie les violations de langage spécifiques à la description de la plate-forme spécifique. Le convertisseur (32) effectue le travail requis pour le chargement des classes et la résolution des références. Le convertisseur effectue la liaison statique des classes, il résout les références symboliques aux classes, méthodes et attributs déjà présents sur la carte. Il répartit le

stockage et crée les structures des données pour représenter les classes, crée les méthodes et attributs statiques ou natifs et initialise les variables statiques.

L'environnement d'exécution de la plate-forme spécifique (Runtime Environment ou RE) comprend la machine virtuelle embarquée ("on-platform") (41) se limitant à un interpréteur, une plate-forme API et les méthodes dites natives (43) ou encore statiques associées. La plate-forme API comprend les API (44) (Application Programming Interface) définissant un ensemble de classes, dites classes système (45), et les conventions d'appel par lesquelles une application accède à l'environnement d'exécution (RE) et aux méthodes statiques (43). Les méthodes statiques (43) exécutent les services d'allocation de mémoire, d'entrée et sortie, et cryptographique de la carte.

L'interpréteur de la machine virtuelle embarquée de la carte (41)(on-platform), sert de support au langage "Java", et lit de façon séquentielle le pseudocode intermédiaire, instruction par instruction. Chaque instruction standard de ce pseudocode intermédiaire est interprétée dans le langage du microprocesseur par l'interpréteur puis exécutée par le microprocesseur. En règle générale, les instructions standards du pseudocode intermédiaire permettent de traiter des fonctions évoluées telle que le traitement arithmétique et la manipulations d'objets. La notion d'objet concerne les objets informatiques tels que listes, tableaux de données ou analogues. Les classes dites clients (46) des applications et les classes systèmes (45) des API sont toutes chargées dans le même espace mémoire et sont gérées par l'intermédiaire d'une table de classe (47).

La machine virtuelle (41) est également chargée de gérer les classes et objets et de faire respecter les séparations ou pare-feu entre les applications afin de permettre un partage sécurisé des données, appelées également attributs, variables ou champs (fields).

Dans le cas d'un objet portable de type carte à puce, une application d'une plate-forme spécifique peut être activée directement par

l'environnement d'exécution (RE) lorsqu'une APDU (Application Protocol Data Unit) de sélection émise par un service ou terminal est reçue par la carte.

Afin de restreindre l'accès aux données entre les parties de code partageant le même espace mémoire, une des méthodes classiques sur une plate-forme classique est basée sur la restriction explicite de visibilité déclarée dans le code source. Suivant la figure 4c, les méthodes (NM1, NM2), respectivement (NM3, NM4), et les attributs (NA1, NA2), respectivement (NA3, NA4) sont encapsulés dans des classes (NCI3), respectivement (NCI2), qui elles-mêmes font partie de paquetages (Paquetage 1), respectivement (Paquetage n) regroupant chacun plusieurs classes. Une classe peut être publique telle que (NCI1 ou NCI2) ou privée telle que (NCI3) par rapport à un paquetage, ce qui implique, dans le dernier cas, que seules les classes du même paquetage peuvent accéder à cette classe. Les méthodes (exemple NM2) et les données (exemple NA1) d'une classe (NCI3) peuvent être privées par rapport à la classe (NM2, NA1), qui elle-même est privée par rapport au paquetage (Paquetage 1), ou publiques par rapport au paquetage (par exemple NM1, NA2) ou à la classe (par exemple NM4, NA4). Cette restriction de la visibilité permet d'obtenir un accès flexible entre les différents ensembles de paquetages (Paquetage 1, Paquetage n) stockés dans le même espace nom, mais présente quelques inconvénients. La plate-forme classique ou spécifique supporte mal la notion de sous-paquetages. Les classes client d'une application de grande taille doivent être réparties entre différents sous-paquetages qui représentent uniquement pour la machine virtuelle des paquetages différents. Pour partager les ressources entre ces sous-paquetages, ces ressources sont nécessairement déclarées publiques, les rendant ainsi visibles de tout autre paquetage. Il est ainsi difficile d'organiser de façon claire le paquetage d'applications différentes pour des applications de grande taille et d'obtenir des pare-feu entre les applications.

Dans le cas de la plate-forme classique, sur une station, le respect de la confidentialité des ressources, telle que déclarée dans les programmes sources, repose sur des vérifications dynamiques. Pour effectuer ces vérifications dynamiques, la machine virtuelle doit posséder toutes les informations concernant les déclarations de restriction d'accès pour chaque classe, méthode ou attribut, ce qui ne peut être possible dans le cas de l'espace mémoire disponible sur la machine virtuelle embarquée (onplatform) utilisant le pseudocode intermédiaire ou byte code prélié de la machine virtuelle hors plate-forme (offplatform). Dans le cas d'une plate-forme spécifique d'un objet portable, les restrictions d'accès peuvent se vérifier statiquement lors de la compilation et peuvent être vérifiées à nouveau par le vérifieur de la partie hors plate-forme. Il est en effet supposé que ce qui est chargé sur la plate-forme spécifique de l'objet portable est correct, car aucune vérification ne peut être effectuée sur la plate-forme spécifique du fait de la perte des informations lors de la phase de conversion. De plus il n'est pas garanti qu'un paquetage ne puisse pas être étendu ou modifié par de nouvelles classes venant de sources étrangères, ce qui peut détruire toute la sécurité basée sur l'utilisation de paquetages privés.

Le deuxième moyen procuré par la machine virtuelle d'une plate-forme classique est la notion de séparation des espaces nom. Avec le schéma de liaison dynamique d'une plate-forme classique, les classes peuvent être chargées, à partir de répertoires du système de fichier local, appelé "ClassPath" préalablement déclaré comme constituant l'emplacement des classes systèmes formant la plate-forme API standard, ou à partir d'autres répertoires du système de fichier local ou de serveurs éloignés. Les classes client d'une application, extérieures au "ClassPath", doivent être chargées par un chargeur de classe spécifiquement programmé. Cette caractéristique est particulièrement utilisée pour le chargement des applications "Java" par des navigateurs habilités "Java". Ainsi, les applications venant de différentes sources, sont chargées par des chargeurs de classes différents. Pour chaque localisateur, communément appelé URL

(Uniform Resource Locator), une instance spécifique de classe "chargeur de classe d'application" est utilisée. Le nom externe d'une classe est l'assemblage <Nom Du Paquetage > + <Nom De La Classe>. Quand une classe est chargée, elle est stockée dans une table de classe interne, et une

5 référence relative au chargeur de classe utilisé pour charger cette classe est ajoutée en préfixe du nom de paquetage de la classe. Le nom de la classe est alors <<Référence du Chargeur de Classe> + <Nom Du Paquetage> + <Nom De La Classe>>. Ainsi des classes déclarées comme appartenant au même paquetage mais chargées par des chargeurs de classes différents ne

10 sont pas perçues par la machine virtuelle comme appartenant au même paquetage.

Lors de la résolution d'une référence, la politique habituelle des chargeurs de classe est de rechercher d'abord dans les classes système et en cas d'échec, de rechercher parmi les fichiers de classe présents à

15 l'emplacement où le chargeur peut charger les classes. Selon ce principe de fonctionnement du chargeur, une application ne peut directement accéder aux classes clients d'une autre application chargée par un chargeur de classes différent. Une application peut accéder à toutes les ressources publiques des classes publiques du ClassPath, mais les classes du

20 ClassPath ne peuvent accéder directement aux classes d'une application bien qu'elles puissent faire référence à des instances de classes clients de l'application en les convertissant en un type public défini dans le "Classpath". Une application ne peut étendre ou modifier des paquetages de classes système du Classpath ou de toutes autres applications chargées par des

25 chargeurs de classes différents. L'utilisation de la séparation des espaces nom en insérant une référence du chargeur de classe dans le nom de la classe, ajoutée au typage complet fourni par le langage "Java", permet de procurer un pare-feu efficace entre les applications. Le mécanisme inné de séparation de nom d'espace permet facilement le chargement de nouvelles

30 applications. Les applications peuvent échanger des objets par l'entremise des classes spécifiquement programmées à cette fin et localisées dans le

"Classpath". Une application peut utiliser un objet venant d'une autre application chargée par un chargeur de classe différent, si cet objet peut être changé en un type public défini dans le "Classpath".

Malheureusement ce mécanisme de séparation de nom, basé sur  
5 une machine virtuelle classique et son processus de liaison dynamique ne peut être effectué sur une plate-forme spécifique. La liaison dynamique ne peut être effectuée par la machine virtuelle d'une plate forme spécifique, celle-ci nécessitant un espace mémoire permettant le stockage de fichier de classe d'une plate-forme "Java" classique, présentant des références non  
10 résolues. Dans une plate-forme spécifique, les classes systèmes des API et les classes clients des applications ne sont pas isolées dans des espaces de nommage particuliers.

L'objet de la présente invention est de proposer une solution procurant les avantages de la séparation de nom dans le cadre de systèmes  
15 embarqués possédant une machine virtuelle en deux parties, le schéma de liaison statique étant effectué par la partie hors plate-forme.

Dans le contexte de systèmes embarqués, tels que par exemple les terminaux de paiement, multi-application, il est nécessaire d'avoir des pare-feu performants entre les applications "Java" fournies par différentes  
20 sources, telles que différents systèmes de paiement (Visa, Mastercard...). Il peut être utile de permettre une coopération flexible entre les applications venant des différentes sources. Le système embarqué doit aussi être facilement mis à jour en téléchargeant de nouvelles applications ou de nouveaux modules utilitaires ou encore des mises à jour sans perturber les  
25 applications déjà chargées.

Un premier but de l'invention est de proposer un procédé de chargement permettant d'obtenir des pare-feu robustes entre les applications tout en autorisant une coopération entre les applications et la possibilité de faire évoluer les applications ou de charger d'autres applications.

30 Ce but est atteint par le fait que le procédé de chargement d'applications sur un système embarqué selon l'invention, comprenant un



environnement d'exécution incluant une machine virtuelle comprenant un interpréteur de langage de type pseudocode intermédiaire, des interfaces de programmation d'application (API), à partir d'une station sur laquelle le code source de l'application est écrit, compilé en pseudocode par un compilateur ,  
 5 vérifié par un vérificateur, converti par un convertisseur et chargé par un chargeur, se caractérise en ce que

- la conversion comprend la réalisation de la liaison statique d'une pluralité d'ensembles de paquetages destinés à être stockés dans le même espace nom sur le système embarqué, appelés modules, en attribuant un  
 10 identificateur à chaque module (MID), et un numéro de référence à chaque classe, à chaque méthode et à chaque attribut encapsulés dans les classes du module,

- la référence à une méthode ou un attribut, dans le pseudocode lié d'un module, étant codée sur trois multiplats constitués par un indicateur  
 15 indiquant la référence à une classe interne ou externe au module, le numéro de la classe et soit le numéro de la méthode soit le numéro de l'attribut,

- les modules chargés sont un ou plusieurs modules d'interface de programmation d'application, appelé Module API, comprenant des classes système ou des modules de services correspondant chacun à une  
 20 application, une référence à une classe externe étant systématiquement interprétée par la machine virtuelle comme une référence à un module d'interface de programmation d'application.

Selon une autre particularité, le chargement des modules sur le système embarqué comprend la mémorisation d'une part d'au moins un  
 25 tableau de représentation des modules, le numéro associé, par le lieu compris entre 0 et n, à un module constituant l'index dudit module dans le tableau, et d'autre part d'une table mémorisant l'association de l'index du tableau de représentation à l'identificateur (MID) dudit module, ledit tableau et la table étant en mémoire programmable non volatile, une référence  
 30 externe à un module externe dans le pseudocode étant interprétée par

l'interpréteur de la machine virtuelle comme constituant un index d'accès au module équivalent à celui du tableau des modules.

Selon une autre particularité, le chargement comprend la mémorisation, pour chaque module, d'un tableau de représentation de ses classes, comprenant une référence à l'index de son module et, pour chaque  
5 classe, un tableau de représentation des attributs et des méthodes.

Selon une autre particularité, les modules sont référencés dans un tableau de modules unique, les classes système sont contenues dans un module API unique, toute référence à une classe externe dans le  
10 pseudocode différente de n sera interprétée par la machine virtuelle comme une référence audit module API.

Selon une autre particularité, les classes étant déclarées publiques, ou en paquetage privé, les attributs et méthodes étant déclarés protégés, en paquetage privée ou en classe privée, la numérotation des classes s'effectue  
15 suivant l'ordre classes publiques puis classes en paquetages privés, la numérotation des attributs ou méthodes est effectuée par le convertisseur suivant l'ordre attribut ou méthode public, protégé, en paquetage privé et en classe privée.

Selon une autre particularité, les classes système sont contenues  
20 dans plusieurs modules API chargeables séparément, le chargeur maintient dans la mémoire non volatile programmable deux tableaux de représentation des modules et deux tables d'association MID/IMi correspondantes, l'un pour les modules API et l'autre pour les modules non-API, le chargeur chargeant les modules dans l'un des deux tableaux selon la nature du module spécifié  
25 dans l'entête de celui-ci, toute référence externe d'un module du tableau de module étant interprétée comme une référence à l'index du module API.

Selon une autre particularité, la liaison statique d'un module est effectuée de telle sorte que la référence à une classe externe à un module non API dans le pseudocode intermédiaire est un index dans un tableau de  
30 l'entête du module, dont chaque entrée est un identificateur (MID) d'un module API référencé, le chargement dudit module sur la plate-forme cible

comprenant le remplacement de ladite référence par le numéro de l'index du module API obtenu à partir de l'identificateur (MID) de la table d'association des modules API.

Un autre but de l'invention est de proposer un système embarqué  
5 correspondant.

Ce but est atteint par le fait que le système embarqué selon l'invention, comprenant une machine virtuelle et une plate-forme API incluant des interfaces de programmation d'application, une mémoire non volatile fixe, une mémoire non volatile programmable ou modifiable, et une mémoire  
10 vive, se caractérise en ce que la mémoire non volatile programmable comprend au moins un module API comprenant des classes système et des modules de services, au moins un tableau de représentation des modules, dans lequel les modules sont indexés et une table associant l'index d'un module du tableau de représentation à l'identificateur dudit module, chaque  
15 module comprenant un tableau de représentation des classes, dans lequel les classes sont indexées et dans lequel chaque classe présente une référence à l'index de son module, chaque classe comprenant un tableau de représentation des attributs et des méthodes, dans lesquels les attributs et méthodes sont indexés, la référence à une méthode ou un attribut étant  
20 codée sur au moins trois multiplats correspondant à une référence à une classe interne ou externe au module, une référence externe au module constituant l'index du module API dans le tableau de module, un numéro de classe correspondant à l'index de la classe dans la table de représentation des classes du module, et un numéro de méthode ou d'attribut  
25 correspondant à l'index de la méthode ou de l'attribut dans le tableau de représentation des méthodes ou attributs de la classe du module.

Selon une autre particularité, le système embarqué comporte des moyens de comparaison du premier multiplat des trois multiplats de codage de référence à une méthode ou à un attribut avec une valeur déterminée n  
30 pour décider s'il s'agit d'une classe interne ou externe.

Selon une autre particularité, le système embarqué comprend un module principal comprenant le programme principal du système.

Selon une autre particularité, les classes sont indexées suivant l'ordre classes publiques puis classes en paquets privés, et les attributs ou méthodes suivant l'ordre attribut ou méthode public, protégé, en  
5 paquetage privé et en classe privée.

Selon une autre particularité, la mémoire non volatile programmable comprend plusieurs modules API comprenant des classes système, deux tableaux de représentation des modules, l'un pour les modules API et l'autre  
10 pour les modules non-API et le module principal, et deux tables d'association MID/IMi correspondant chacune à un tableau de représentation des modules.,

Selon une autre particularité, le système embarqué comprend une classe gestionnaire d'accès "Access manager" d'un module API comprenant  
15 une méthode permettant de créer une instance d'un module de service, par l'intermédiaire du module principal, ladite classe présentant une protection lui interdisant d'avoir plus d'une instance.

Un autre but de l'invention est de proposer un procédé d'exécution d'une application présente sur un système embarqué multi-application.

20 Ce but est atteint par le fait que le procédé d'exécution d'une application d'un système embarqué multi-application, comprenant un environnement d'exécution incluant une machine virtuelle comprenant un interpréteur de langage de type pseudocode intermédiaire, et des interfaces de programmation d'applications (API), se caractérise en ce que, lors de  
25 l'exécution du pseudocode intermédiaire d'un module de service, correspondant à une application, référencée dans un tableau de module, la référence à une méthode ou un attribut dans le pseudocode, codée sur au moins trois multipliants correspondant à une référence à une classe interne ou externe au module, un numéro de classe et un numéro de méthode ou  
30 d'attribut, une référence externe au module est interprétée par la machine

virtuelle comme une référence à l'index d'un module API du tableau du ou des modules API.

Selon une autre particularité, sur réception d'une demande d'exécution d'un module de service présentant un identificateur, l'environnement d'exécution accède à la classe d'entrée d'un module principal comprenant le programme principal du système, le module principal installe une instance d'une classe spéciale "Access Manager", d'un module API, gérant l'accès à un module de service et utilise une méthode de cette classe permettant de créer une instance de la classe d'entrée du module de service demandée, par l'intermédiaire d'une table d'association de l'identificateur à l'index du module dans un tableau dans lequel le module est référencé, l'instance étant retournée par la méthode au programme principal.

D'autres particularités et avantages de la présente invention apparaîtront plus clairement à la lecture de la description ci-après faite en référence aux dessins annexés dans lesquels :

- la figure 1 représente de façon schématique les différents éléments nécessaires pour le chargement d'un objet portable selon un premier mode de réalisation ;
- la figure 2 représente de façon schématique les différents éléments nécessaires pour le chargement d'un objet portable selon un deuxième mode de réalisation ;
- la figure 3 représente la représentation interne d'un module ;
- la figure 4a représente le schéma classique d'une carte à puce ;
- la figure 4b représente le système nécessaire à la constitution d'une machine virtuelle embarquée sur une carte à puce selon l'art antérieur;
- la figure 4c représente la structure des classes d'une application.

Le procédé sera décrit, en liaison avec les figures 1 à 3, de manière non limitative, dans le cas de la mise en œuvre de l'invention dans un système embarqué, par exemple de type spécifique constitué par une carte à puce ou un objet portable similaire. La désignation byte code ou

programme de type byte code recouvre tout pseudocode ou programme intermédiaire.

L'objet portable constitue par exemple une carte à puce et présente une structure similaire de celle décrite précédemment en référence aux figures 4a et 4b, et comprend notamment une mémoire RAM, ROM et EEPROM. La plate-forme spécifique (60) et une station classique (80) sont représentées sur la figure 1. La plate-forme spécifique (60) possède en ROM, un environnement d'exécution (RE) comprenant des API (62) et une machine virtuelle (61) embarquée ("onplatform"). La plate-forme spécifique (60) est représentée sur la figure 1, comme comprenant l'ensemble des mémoires ROM et EEPROM. Il est à noter que la plate-forme spécifique (60) désigne plus particulièrement l'environnement d'exécution (RE) et les éléments présents en mémoire EEPROM. L'objet portable possède en ROM un système d'exploitation (Operating System) (63). Les API (62), présentes en mémoire ROM, constituent les API de base de la plate-forme API, chargées avec la machine virtuelle embarquée pour le fonctionnement de celle-ci.

La partie (90) hors objet portable de la machine virtuelle comprend un vérifieur (91) de pseudocode intermédiaire, un convertisseur (92) et éventuellement un signeur (94). Le signeur délivre une signature pour valider le passage par le vérifieur et le convertisseur. Cette signature sera vérifiée par l'objet portable au moment du chargement. Le chargement en EEPROM d'applications ou de nouvelles API pour compléter les API de base s'effectue par un chargeur qui peut être composé de deux parties, une partie hors objet portable pouvant être installée dans la machine virtuelle hors objet portable (90), appelée téléchargeur (93), et une partie sur la plate-forme spécifique, appelée chargeur (68).

Selon un premier mode de réalisation, la plate-forme spécifique (60) comprend deux modules spéciaux, un module API (65) et un module principal (66). Les autres modules sont appelés modules de services (67). Chaque module correspond à un ensemble de paquetages qui sera stocké

dans le même espace nom. La plate forme API désigne les API de base (62)  
 et l'ensemble des classes système qui définissent le module API (65) ou  
 module de la plate-forme API. Le module principal comprend la classe  
 principale définissant le programme principal. Chaque module, excepté le  
 5 module API (65), possède une classe unique, particulière, appelée "Entry  
 Class", qui constitue le point d'accès au module. Pour le module principal,  
 cette "Entry Class" est la classe principale (CP), celle qui contient une  
 méthode statique appelée "main". Pour les modules de services, c'est une  
 classe avec seulement un constructeur sans paramètres et implémentant  
 10 une interface publique spéciale, appelée "service" définie dans la plate-forme  
 API. Le chargement d'une application correspond au chargement d'un  
 module de service. Chaque module reçoit un identificateur spécifique. Un tel  
 identificateur, qui est appelé MID, peut par exemple être un nombre, une  
 chaîne de caractères, ou un tableau. A titre d'exemple, l'identificateur est  
 15 une chaîne de caractères.

Quand ils sont chargés dans la plate-forme, par des mécanismes de  
 téléchargement distincts de la machine virtuelle de la plate-forme spécifique,  
 les modules reçoivent un nombre, compris entre 0 et n. Ainsi, selon cette  
 convention, n+1 modules au plus peuvent être présents sur la plate-forme  
 20 spécifique. Le téléchargeur (93) de module avec le chargeur (68) maintient,  
 lors du chargement de nouveaux modules de service, un tableau (TRM) (69)  
 de représentation des modules. Le numéro associé à un module est l'index  
 (IM) de ce module dans le tableau. Le chargeur (68) maintient également  
 une table (70) associant l'index (IM) à l'identificateur (MID) de chaque  
 25 module. Le module API reçoit systématiquement pour index IM<sub>0</sub> le numéro 0,  
 et le module principal pour index IM<sub>1</sub> le numéro 1. L'entête (header) de  
 chaque module comprend un indicateur permettant au chargeur de  
 déterminer la nature du module, "principal", modules "de service" ou module  
 "API".

30 Le chargeur (68) ne peut charger que les modules autorisés à  
 résider sur l'objet portable, c'est à dire uniquement les modules présentant

une signature connue de l'objet portable. Le chargeur (68) comporte donc des moyens de vérifier la signature d'un module reçu, de la comparer avec la signature connue de l'objet portable et en cas de comparaison négative de bloquer le chargement.

5 De manière classique, tel que défini dans l'art antérieur cité précédemment, le programme source (81) d'une application est écrit puis compilé par un compilateur (82) et ensuite vérifié par le vérifieur (91).

La liaison statique, réalisée dans le convertisseur (92) par un composant dit lieu (linker) du convertisseur, va résoudre des références  
10 symboliques en attribuant

- un numéro (NCI) à chaque classe d'un module,
- un numéro (NM) pour chaque méthode dans une classe et
- un numéro (NA) pour chaque attribut dans une classe.

Chacun de ces numéros (NCI, NM, NA) est compris entre 0 et n et  
15 peut ainsi être représenté sur un byte ou multiplet. A titre d'exemple, chacun de ces nombres sera compris entre 0 et 255 ( $n=255$ ). La référence à une méthode ou un attribut d'une classe sera ainsi codée dans le pseudocode lié des méthodes du module sur deux multiplets (bytes) ou octets. Le pseudocode contiendra ces deux multiplets < NCI> pour la classe et < NA>  
20 pour un attribut ou <NM> pour une méthode.

Suivant la figure 3, la représentation interne d'un module API (65), d'un module principal (66) ou d'un module de service (67), contiendra un tableau (TRC) de représentation de classes ; le numéro (NCI) associé par le lieu, hors du système embarqué, à chaque classe est l'index (ICi) de la  
25 représentation de cette classe dans le tableau (TRC). Chaque classe présente également une référence à l'index (IMi) de son module. De la même manière, la représentation de chaque classe contient un tableau des représentations de méthodes (TRMe) et un tableau de représentation des attributs (TRA) appartenant à la classe. Le numéro (NM), associé par le lieu,  
30 hors du système embarqué, à chaque méthode est l'index (IMi), de la représentation de cette méthode dans le tableau (TRMe), et le numéro (NA),



associé par le lieu, hors du système embarqué, à chaque attribut est l'index (IAi), de la représentation de cet attribut dans le tableau (TRA).

A titre d'exemple, nous voulons qu'un module puisse référer uniquement à ses propres classes et aux classes systèmes de la plate-forme API, les classes système correspondant aux classes du "ClassPath" d'une plate-forme classique. Selon l'invention, pour permettre la distinction entre une référence à une classe interne au module et la référence à une classe système (ou externe au module), un indicateur interne (II) ou externe (IE) est ajouté à la référence à une méthode ou à un attribut. La référence résolue est alors codée sur trois multiplats : <IE/II> <NCI> <NM> ou <IE/II> <NCI> <NA>

Selon une convention posée, pour la valeur n du premier multiplat <IE/II> la valeur 255 d'après notre exemple, correspond à une référence interne <II> au module et toute autre valeur pour le premier multiplat correspond à une référence externe <IE> au module.

Le lieu du convertisseur (92) de la machine virtuelle hors objet portable (90), relie en premier le module API (65), qui ne possède pas de références externes <IE> dans son pseudocode, et produit une implantation ou agencement, correspondant à un plan de noms symboliques de ses classes et de leurs méthodes et attributs. Lors de la mise en liaison des autres modules, cette implantation sera utilisée pour établir les références externes à des classes systèmes du module API (65).

Suivant notre convention des multiplats (bytes) encodant les références, il peut y avoir au plus 256 (n+1) classes dans le module API et 256 classes dans chaque module supplémentaire.

Lors de l'exécution d'un module de service, quand la machine virtuelle (61) trouve une référence à une méthode <NM> ou un attribut <NA> dans le pseudocode, en connaissant la classe <NCI> où se trouve cette référence, elle peut retrouver directement l'index <IMi> du module concerné, celui-ci correspondant à la référence externe (IE) ou interne (II). Toute référence externe <IE> dans le pseudo code d'un module de service sera

systématiquement interprétée par la machine virtuelle comme une référence  
 au module API. Un module de service ou le module principal ne peut pas  
 référer aux classes de tout autre module excepté celles du module API. Les  
 classes systèmes de ce module API ne peuvent pas référer aux classes d'un  
 5 module de service ou du module principal. La référence interne à une classe  
 d'un module, correspondant à la valeur  $n$  pour le premier multiplet, ne  
 nécessite aucune connaissance a priori de l'espace nom qui sera attribué au  
 module. Le fait de ne pas définir a priori d'espace de nommage fixe lors de la  
 phase de conversion permet d'accélérer la résolution des références et de  
 10 déterminer l'espace de nommage d'un module lors du chargement,  
 postérieurement à la phase de conversion. La machine virtuelle, lors de  
 l'interprétation d'une référence à un attribut ou à une méthode dans le  
 pseudocode, utilise les trois index  $\langle IE/II \rangle$   $\langle NCI \rangle$   $\langle NM \rangle$  ou  $\langle IE/II \rangle$   $\langle NCI \rangle$   
 $\langle NA \rangle$  en cascade. L'espace mémoire du module étant déterminé, l'index  
 15  $\langle NCI \rangle$  détermine l'entrée désirée dans le tableau des classes (TRM) du  
 module, puis le dernier index  $\langle NM \rangle$  ou  $\langle NA \rangle$  donne l'entrée désirée dans le  
 tableau des méthodes (TRMe) ou le tableau des attributs (TRA).

Le module API comprend une classe spéciale (64), appelée classe  
 "gestionnaire d'accès" ou "Access Manager", qui comprend une méthode  
 20 native (getServiceInstance) dont le rôle est de rendre un objet instance de la  
 classe d'entrée du module de service demandé, à partir de l'identificateur  
 (MID) du module. Cette méthode utilise la table (70) d'association MID/Imi  
 pour connaître l'index du module demandé dans le tableau de module (69)  
 puis crée une instance de la classe d'entrée de ce module, instance qui est  
 25 retournée par la méthode. Selon l'invention la classe "Access Manager" (64)  
 est protégée par construction par une méthode consistant à interdire que  
 cette classe ait plus d'une instance. Cette méthode (getServiceInstance)  
 appartient au programme principal contenu dans le module principal. Le  
 module principal qui sera activé en premier à utilisation de l'objet portable  
 30 crée une instance et une seule de la classe "Access Manager", ce qui lui

permet d'utiliser la méthode `getServiceInstance`, mais interdit à tout autre service de créer une autre instance pour utiliser cette méthode.

En fonctionnement, de la même façon que sur une plate-forme classique, l'environnement d'exécution (RE) accède à la classe d'entrée (EC) du module principal et active sa méthode d'entrée (`main`). Le module principal, étant le premier activé, procède à l'installation d'une instance de la classe "Access manager" avant que tout autre service le fasse, puisque pour activer d'autres services, le module principal doit déjà posséder une telle instance de la classe accès.

Ce simple dispositif permet de reproduire l'effet de protection lié au concept d'espace de nommage d'une plate-forme classique. Le simple fait de charger un module de service dans le tableau des modules et que la présence dans le pseudocode de toutes référence externe soit interprétée par la machine virtuelle comme une référence au module API, rend ce module complètement inaccessible directement par les autres modules, créant ainsi un pare-feu total.

Ce premier mode de réalisation permet d'apporter les avantages de pare-feu procuré par la séparation des espaces nom dans le contexte d'une machine virtuelle en deux parties. Toutefois ce mode de réalisation se révèle peu flexible et présente deux inconvénients.

Premièrement, il empêche toute modification ou extension des classes systèmes avec des modules déjà préliés. Une architecture "Java" classique permet de modifier et d'étendre les classes de la plate-forme API sans avoir d'impact sur les classes déjà compilées de modules supplémentaires. Mais dans le mode de réalisation décrit précédemment, toute modification des classes système, même invisible pour des modules étrangers, modifierait l'agencement de la plate-forme API et nécessiterait de modifier le pseudocode prélié de chaque module déjà lié avec une version antérieure de l'agencement et en conséquence l'interpréteur.

Deuxièmement, les modules préliés sont supposés être portables entre les différentes plates-formes ou terminaux embarqués, ce qui impose

que chacune de ces plates-formes doit avoir le même agencement que la plate-forme API, ce qui interdit l'utilisation de toute extension propriétaire.

Afin de remédier partiellement à ces inconvénients, une variante du premier mode de réalisation, consiste à imposer que, dans la numérotation de l'implantation, les classes publiques viennent en premier avant les classes en paquetage privé. De plus, les méthodes publiques ou les attributs publics viennent avant ceux qui sont protégés et ceux qui sont en paquets privés et en classes privées. Ceci permet d'ajouter librement de nouvelles classes publiques dans le module API (65).

La figure 2 représente un deuxième mode de réalisation permettant l'évolution de la plate-forme API. La plate-forme API est constituée de plusieurs modules API (100) pouvant être chargés séparément, au lieu d'être constituée d'un module API unique.

Dans ce mode de réalisation, le téléchargeur (93) et la machine virtuelle partagent deux tableaux de modules et deux tables d'association MID/index au lieu d'un de chaque, un tableau (101) et une table d'association (102) pour les modules API et un tableau (103) et une table d'association (104) pour les modules non-API, correspondant au module de service (67) et au module principal (66).

Chaque module présente dans son entête un indicateur indiquant sa nature "Service" ou "API" permettant au chargeur de charger le module dans le tableau (101) de modules API ou dans le tableau (103) de modules non-API. Cet indicateur est placé dans l'entête du module lors de la phase de compilation par le convertisseur.

Le pare-feu constitué par la séparation de l'espace nom est présent uniquement entre les modules non-API. Toute référence externe à un module de service sera interprétée par l'interpréteur de la machine virtuelle embarquée comme un index du tableau du module API.

Les modules non-API seront numérotés de 0 jusqu'à 255 au plus, dans l'exemple où  $n=255$ . 0 est par exemple l'index du module principal (66). Les modules API (100) seront numérotés de 0 à 254 au plus, 0 étant par

exemple l'index d'un module dit API primaire, qui contient toutes les méthodes natives. Conformément à la convention décrite précédemment, ceci permet au plus, 255 (n) modules différents dans la plate-forme API. La référence à une méthode ou attribut dans le pseudocode est :

5                    << IE/II> <NCI> < NM/NA>>

La valeur 255 (n) pour le premier multiplet (byte) indiquera, comme dans le premier mode de réalisation, une référence interne au module. Chaque valeur différente de 255 indiquera une référence externe à un module spécifique (100) du tableau de module API de la plate-forme API.

10                  Après la réalisation de la liaison par le lieur (92) hors plate-forme, le pseudocode d'un module comporte une entête présentant un tableau de modules référencés utilisé pour lier le module courant. Ce tableau de modules référencés comprend au plus 255 entrées, chaque entrée correspondant à l'identificateur (MID) d'un module API (100). Le premier  
15                  multiplet (byte) d'une référence externe dans le pseudocode sera alors un index dans ce tableau. Lors du chargement sur la plate-forme d'un module non API (67, 66), les numéros d'index associés à des modules API (100) seront connus et ainsi chaque premier octet d'une référence externe sera remplacé par le numéro d'index associé au module API référé en utilisant la  
20                  table (102) d'association MID/IMi des modules API (100). Ce remplacement est la seule opération de liaison réalisée sur la plate-forme spécifique, par le chargeur (68), la table (102) d'association MID/IMi étant utilisée uniquement pour réaliser cette opération de liaison.

A titre d'exemple, supposons que dans le pseudocode d'un module  
25                  de service "TEST", nous avons la référence à un numéro de méthode 5 du numéro de classe 7 d'un module API dont l'identificateur (MID) est "F00". Supposons de plus que "F00" est l'identificateur du quatrième module API externe trouvé référencé dans le module de service "TEST". La référence dans le pseudocode est alors constituée par les trois valeurs suivantes : 3, 7,

30                  5.

Le numéro 3 correspond au quatrième index dans le tableau de modules référencés présent dans l'entête du module, venant en tête du pseudocode, la valeur de cette entrée étant l'identificateur (MID) "FOO". Supposons que lors du chargement du module API "FOO, l'index interne 34  
 5 lui ait été attribué sur la plate-forme cible dans la table d'association (102) des modules API. Alors, le chargeur (68), à l'aide de la table d'association (102) modifie la référence dans le pseudocode du module de service "TEST" pour devenir : 34, 7, 5.

Lors de l'exécution du pseudocode d'un module, une référence  
 10 externe au module est systématiquement interprétée par la machine virtuelle comme une entrée dans la table de module API. Les modules du tableau de module non API restent invisibles les uns des autres ainsi que par rapport aux modules API. Ce simple dispositif permet de reproduire l'effet de protection lié au concept d'espace de nommage d'une plate-forme classique.  
 15 Le simple fait de charger un module dans le tableau de module non API le rend complètement inaccessible directement par les autres modules, créant ainsi un pare-feu total.

Le tableau (101) de modules API comprend un module spécifique (105), appelé module "API Access" qui comprend une méthode native  
 20 (getServiceInstance) dans une classe "gestionnaire d'accès" ou "Access Manager" dont le rôle est de rendre un objet instance de la classe d'entrée du module de service demandé. Cette méthode utilise la table (104) d'association MID/IMi pour connaître l'index du module de service demandé dans le tableau (103) de modules non-API puis crée une instance de la  
 25 classe d'entrée de ce module qui est retournée par la méthode au programme principal. La politique de sécurité préconisée est de faire de la classe "Access Manager" une classe protégée dont le constructeur et les méthodes sont déclarés protégés. De plus, le module "API Access" (105) comprend une protection consistant à interdire que la classe "Access  
 30 Manager" ait plus d'une instance. Cette méthode est réservée au programme principal contenu dans le module principal (66). Le module principal qui est

activé en premier crée une instance du module Access manager, ce qui lui permet d'utiliser la méthode getServiceInstance, mais interdit à tout autre service de créer une autre instance pour utiliser cette méthode. Ainsi le module principal pourra créer des instances de services.

5 Plusieurs méthodes peuvent être utilisées pour obtenir cette protection consistant à interdire que la classe "Access manager" n'ait qu'une seule instance. Le constructeur de la classe peut par exemple bloquer la demande de création d'instance lorsqu'il en existe déjà une et lance une exception de sécurité. En fonctionnement, l'environnement d'exécution (RE)  
10 accède à la classe d'entrée du module principal (66) et active sa méthode d'entrée (main). Le module principal étant le premier activé, procède à l'installation d'une instance de la classe "Access Manager" du module Access avant que tout autre service le fasse.

Afin de permettre à un module de service d'activer un autre module  
15 de service, cette politique stricte de sécurité peut être modifiée en ajoutant à la classe "Access Manager" du module API Access (105) des classes publiques permettant à tout module d'y faire des requêtes. Ces requêtes seront traitées et contrôlées par l'unique instance créée par le module principal. Ces classes publiques comprennent notamment une méthode  
20 statique permettant d'obtenir l'unique instance. Un module ayant accès à l'objet instance de la classe "Access Manager" pourra activer un autre module de service et l'utiliser, mais il ne pourra pas référencer directement ses classes, ses méthodes ou ses attributs sans être repéré par la machine virtuelle, étant donné que toute référence externe dans le pseudocode est  
25 une référence interne au module ou une référence externe à un module API.

Pour une réalisation simple de cette solution, il est nécessaire de ne pas avoir de références circulaires parmi les modules API. En conséquence, la fermeture transitive de la relation "réfère à"("refers to") doit être un ordre strict partiel sur un jeu de modules. Il est ainsi possible de concevoir dans le  
30 lieu du convertisseur (92) une stratégie simple pour lier et produire l'agencement des modules API en traitant en premier les éléments

minimums non encore liés. Il est possible de suivre la même stratégie basée sur un ordre partiel pour le téléchargement des modules API, de telle sorte que lors du téléchargement d'un module M, tous les modules auquel il se réfère aient déjà été téléchargés et qu'un numéro leur aient été assigné.

- 5 L'assignation de l'index interne sur la plate-forme cible se fait par le chargeur (68) de module en assignant l'index n-1 à l'API d'ordre n. Un module API ne peut référer à un autre API module d'index supérieur.

L'utilisation de ce système de double tableau de modules (101, 103) et de table d'association (102, 104), permet de remplacer facilement un

- 10 module API unique par plusieurs modules API chargeables séparément. Le remplacement d'un module API unique par plusieurs modules API permet d'étendre la plate-forme API avec de nouveau modules, sans modifier l'assemblage des modules déjà chargés, sans changer la sécurité offerte par les pare-feu. Bien entendu, ces deux modes de réalisation ne sont pas
- 15 compatibles ; les modules doivent être préliés spécifiquement pour l'un ou l'autre des modes de réalisation, le pseudocode relatif à un des modes de réalisation n'étant pas portable sur une plate-forme implémentant l'autre mode de réalisation. De plus, l'interpréteur de la machine virtuelle diffère d'un mode de réalisation à l'autre. Dans le premier mode de réalisation, la
- 20 machine virtuelle ne manipule qu'un seul tableau et une table d'association : le premier multiplet d'une référence sera interprété par la machine virtuelle comme une référence interne pour toute valeur égale à n et comme une référence externe à l'unique module API pour toute valeur différente de n. Dans le second mode de réalisation, la machine virtuelle manipule deux
- 25 tableaux et deux tables d'association : le premier multiplet d'une référence dans le pseudocode sera interprété par la machine virtuelle comme une référence interne au module pour toute valeur égale à n et toute valeur différente de n sera prise directement comme index dans le tableau de module API. Dans les deux modes de réalisation l'interpréteur de la machine
- 30 virtuelle comprend des moyens de comparaison du premier multiplet des trois multiplets de codage d'une référence à une méthode ou à un attribut



avec une valeur déterminée  $n$  pour décider s'il s'agit d'une classe interne ou externe au module. La numérotation des modules API peut être déterminée au moment du chargement pour fixer définitivement et de manière très simple les références externes dans le pseudocode.

- 5            Les mêmes mécanismes sont utilisés pour manier les deux types de modules, bien que la façon dont ils sont utilisés et la sécurité procurée soient tout à fait différentes. Tout module peut accéder librement aux modules API puisque leurs classes sont des classes système. L'utilisation de l'approche modulaire est utilisée avec les modules services pour procurer un pare-feu
- 10 rigoureux pour protéger ces modules de tout accès direct.

Le procédé selon l'invention peut être réalisé sur tous types d'objet portable présentant de faibles ressources, tel que par exemple, 16 Ko de mémoire ROM, 8ko de mémoire EEPROM et 256 k de mémoire RAM.

- D'autres modifications à la portée de l'homme de métier font
- 15 également partie de l'esprit de l'invention.

## **REVENDECATIONS**

1. Procédé de chargement d'applications sur un système embarqué comprenant un environnement d'exécution incluant une machine virtuelle  
5 comprenant un interpréteur de langage de type pseudocode intermédiaire, des interfaces de programmation d'application (API), à partir d'une station sur laquelle le code source de l'application est écrit, compilé en pseudocode par un compilateur (82), vérifié par un vérificateur (91), converti par un convertisseur (92), et chargé par un chargeur (93, 68), caractérisé en ce que
- 10       - la conversion comprend la réalisation de la liaison statique d'une pluralité d'ensembles de paquetages destinés à être stockés dans le même espace nom sur le système embarqué, appelés modules, en attribuant un identificateur à chaque module (MID), et un numéro de référence à chaque classe (NCI), à chaque méthode (NM) et à chaque attribut (NA) encapsulés
- 15 dans les classes du module,
- la référence à une méthode ou un attribut, dans le pseudocode lié d'un module, étant codée sur trois multiplats constitués par un indicateur indiquant la référence à une classe interne (II) ou externe (IE) au module, le numéro de la classe (NCI) et soit le numéro de la méthode (NM) soit le
- 20 numéro de l'attribut (NA),
- les modules sont un ou plusieurs modules d'interface de programmation d'application comprenant des classes système ou des modules de services correspondant chacun à une application, une référence (IE) à une classe externe étant systématiquement interprétée par la machine
- 25 virtuelle comme une référence à un module d'interface de programmation d'application.
2. Procédé de chargement d'applications sur un système embarqué selon la revendication 1, caractérisé en ce que le chargement des modules sur le système embarqué comprend la mémorisation d'une part d'au moins
- 30 un tableau (69, 101, 103) de représentation des modules, le numéro (IMi) associé, par le lieu compris entre 0 et n, à un module constituant l'index

dudit module dans le tableau, et d'autre part d'une table (70, 102, 104) mémorisant l'association de l'index du tableau de représentation à l'identificateur (MID) dudit module, ledit tableau et la table étant en mémoire programmable non volatile, une référence externe (IE) à un module externe  
 5 dans le pseudocode étant interprétée par l'interpréteur de la machine virtuelle comme constituant un index d'accès au module équivalent à celui du tableau des modules.

3. Procédé de chargement d'applications sur un système embarqué selon la revendication 2, caractérisé en ce que le chargement comprend la  
 10 mémorisation, pour chaque module, d'un tableau de représentation (TRC) de ses classes, comprenant une référence à l'index de son module et, pour chaque classe, un tableau de représentation (TRA) des attributs et des méthodes (TRMe).

4. Procédé de chargement d'applications dans un système  
 15 embarqué selon la revendication 2, caractérisé en ce que les modules sont référencés dans un tableau de modules unique, les classes système sont contenues dans un module API unique, toute référence à une classe externe dans le pseudocode différente de n sera interprétée par la machine virtuelle comme une référence audit module API.

20 5. Procédé de chargement d'applications dans un système embarqué selon la revendication 4, caractérisé en ce que les classes étant déclarées publiques, ou en paquetage privé, les attributs et méthodes étant déclarés protégés, en paquetage privée ou en classe privée, la numérotation des classes s'effectue suivant l'ordre classes publiques puis classes en  
 25 paquetages privés, la numérotation des attributs ou méthodes est effectuée par le convertisseur (92) suivant l'ordre attribut ou méthode public, protégé, en paquetage privé et en classe privée.

6. Procédé de chargement d'applications sur un système embarqué selon la revendication 2, caractérisé en ce que les classes système sont  
 30 contenues dans plusieurs modules API chargeables séparément, le chargeur (68) maintient dans la mémoire non volatile programmable deux

tableaux de représentation des modules et deux tables d'association MID/IMI correspondantes, l'un pour les modules API et l'autre pour les modules non-API, le chargeur chargeant les modules dans l'un des deux tableaux selon la nature du module spécifié dans l'entête de celui-ci, toute référence externe d'un module du tableau de module étant interprétée comme une référence à l'index du module API.

7. Procédé de chargement d'applications sur un système embarqué selon la revendication 6, caractérisé en ce que la liaison statique d'un module est effectuée de telle sorte que la référence à une classe externe à un module non API dans le pseudocode intermédiaire est un index dans un tableau de l'entête du module, dont chaque entrée est un identificateur (MID) d'un module API référencé, le chargement dudit module sur la plate-forme cible comprenant le remplacement de ladite référence par le numéro de l'index du module API obtenu à partir de l'identificateur (MID) de la table d'association MID/IMI des modules API.

8. Système embarqué comprenant une machine virtuelle et une plate-forme API incluant des interfaces de programmation d'application, une mémoire non volatile fixe, une mémoire non volatile programmable ou modifiable, et une mémoire vive, caractérisé en ce que la mémoire non volatile programmable comprend au moins un module API comprenant des classes système et des modules de services, au moins un tableau de représentation des modules (TRM), dans lequel les modules sont indexés et une table (70, 104) associant l'index (IM) d'un module du tableau de représentation à l'identificateur (MID) dudit module, chaque module comprenant un tableau de représentation des classes (TRC), dans lequel les classes sont indexées et dans lequel chaque classe présente une référence à l'index (IM) de son module, chaque classe comprenant un tableau de représentation des attributs (TRA) et des méthodes (TRMe), dans lesquels les attributs et méthodes sont indexés, la référence à une méthode ou un attribut étant codée sur au moins trois multiplats (bytes) correspondant à une référence à une classe interne (II) ou externe (IE) au module, une référence

externe au module constituant l'index du module API dans le tableau de module, un numéro de classe (NCI) correspondant à l'index de la classe dans la table de représentation des classes du module, et un numéro de méthode (NM) ou d'attribut (NA) correspondant à l'index de la méthode ou de l'attribut dans le tableau de représentation des méthodes ou attributs de la classe du module.

9. Système embarqué selon la revendication 8, caractérisé en ce qu'il comporte des moyens de comparaison du premier multiplet des trois multiplets de codage de référence à une méthode ou à un attribut avec une valeur déterminée n pour décider s'il s'agit d'une classe interne ou externe.

10. Système embarqué selon la revendication 8, caractérisé en ce qu'il comprend un module principal comprenant le programme principal du système.

11. Système embarqué selon la revendication 8, caractérisé en ce que les classes sont indexées suivant l'ordre classes publiques puis classes en paquets privés, et les attributs ou méthodes suivant l'ordre attribut ou méthode public, protégé, en paquetage privé et en classe privée.

12. Système embarqué selon la revendication 11, caractérisé en ce que la mémoire non volatile programmable comprend plusieurs modules API comprenant des classes système, deux tableaux de représentation (101, 103) des modules, l'un pour les modules API et l'autre pour les modules non-API, et le module principal, et deux tables (102, 104) d'association MID/IMI correspondant chacune à un tableau de représentation des modules.

13. Système embarqué selon la revendication 10, caractérisé en ce qu'il comprend une classe gestionnaire d'accès "Access manager" d'un module API (105) comprenant une méthode permettant de créer une instance d'un module de service, par l'intermédiaire du module principal, ladite classe présentant une protection lui interdisant d'avoir plus d'une instance.

14. Procédé d'exécution d'une application d'un système embarqué multi-application, comprenant un environnement d'exécution incluant une

machine virtuelle comprenant un interpréteur de langage de type pseudocode intermédiaire, et des interfaces de programmation d'application (API), caractérisé en ce que, lors de l'exécution du pseudocode intermédiaire d'un module de service, correspondant à une application, référencée dans un tableau de module, la référence à une méthode ou un attribut dans le pseudocode, codée sur au moins trois multiplats (bytes) correspondant à une référence à une classe interne (II) ou externe (IE) au module, un numéro de classe (NCI) et un numéro de méthode (NM) ou d'attribut (NA), une référence externe au module est interprétée par la machine virtuelle comme une référence à l'index d'un module API du tableau du ou des modules API .

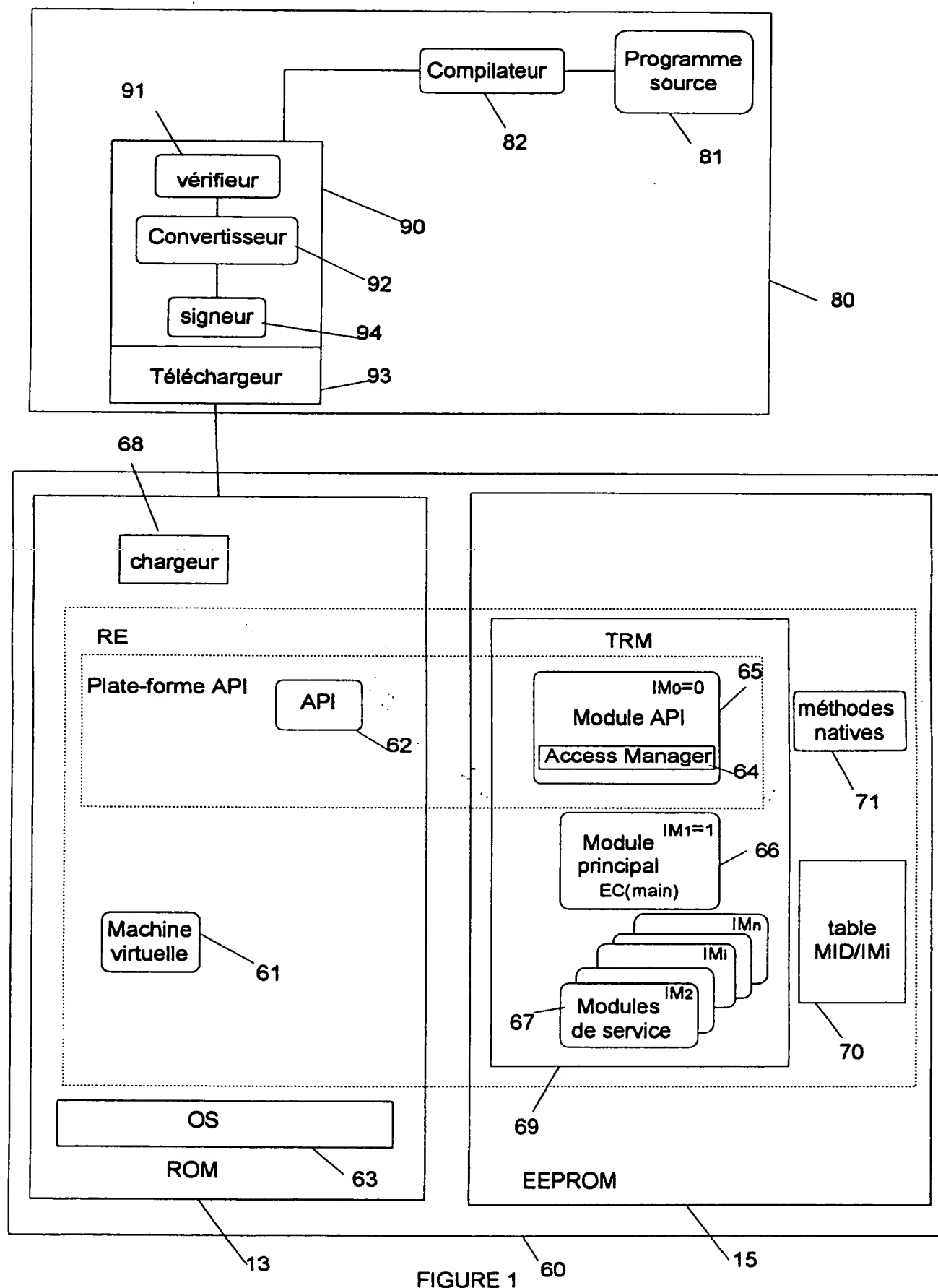
15. Procédé d'exécution d'une application d'un système embarqué multi-application selon la revendication 14, caractérisé en ce que, sur réception d'une demande d'exécution d'un module de service présentant un identificateur (MID), l'environnement d'exécution accède à la classe d'entrée d'un module principal comprenant le programme principal du système, le module principal installe une instance d'une classe spéciale "Access Manager" d'un module API, gérant l'accès à un module de service, et utilise une méthode de cette classe permettant de créer une instance de la classe d'entrée du module de service demandée, par l'intermédiaire d'une table d'association de l'identificateur à l'index du module dans un tableau dans lequel le module est référencé, l'instance étant retournée par la méthode au programme principal.

**ABREGE**

**Procédé de chargement d'applications dans un système embarqué multi-application muni de ressources de traitement de données, système embarqué correspondant, et procédé d'exécution d'une application d'un système embarqué correspondant**

La présente invention concerne un procédé de chargement d'applications dans un système embarqué multi-application, le système embarqué correspondant, et le procédé d'exécution d'une application du système embarqué. Le procédé de chargement d'applications sur un système embarqué comprenant un environnement d'exécution incluant une machine virtuelle comprenant un interpréteur de langage de type pseudocode intermédiaire, des interfaces de programmation d'application (API), à partir d'une station sur laquelle le code source de l'application est écrit, compilé en pseudocode, vérifié et converti, se caractérise en ce que la conversion comprend la réalisation de la liaison statique d'une pluralité d'ensemble de paquetages destinés à être stockés dans le même espace nom sur le système embarqué, appelés modules, et constituant un module d'interface de programmation d'application ou un module de service correspondant à une application, et consiste à attribuer un identificateur à chaque module (MID), et un numéro de référence à chaque classe (NCI), à chaque méthode (NM) et à chaque attribut (NA). La référence à une méthode ou un attribut, dans le pseudocode lié d'un module est codée sur trois multiplets constitués par un indicateur indiquant la référence à une classe interne (II) ou externe (IE) au module, le numéro de la classe (NCI) et soit le numéro de la méthode (NM) soit le numéro de l'attribut (NA), une référence (IE) à une classe externe étant systématiquement interprétée par la machine virtuelle comme une référence à un module d'interface de programmation d'application.

## PL 1/4





# PL 2/4

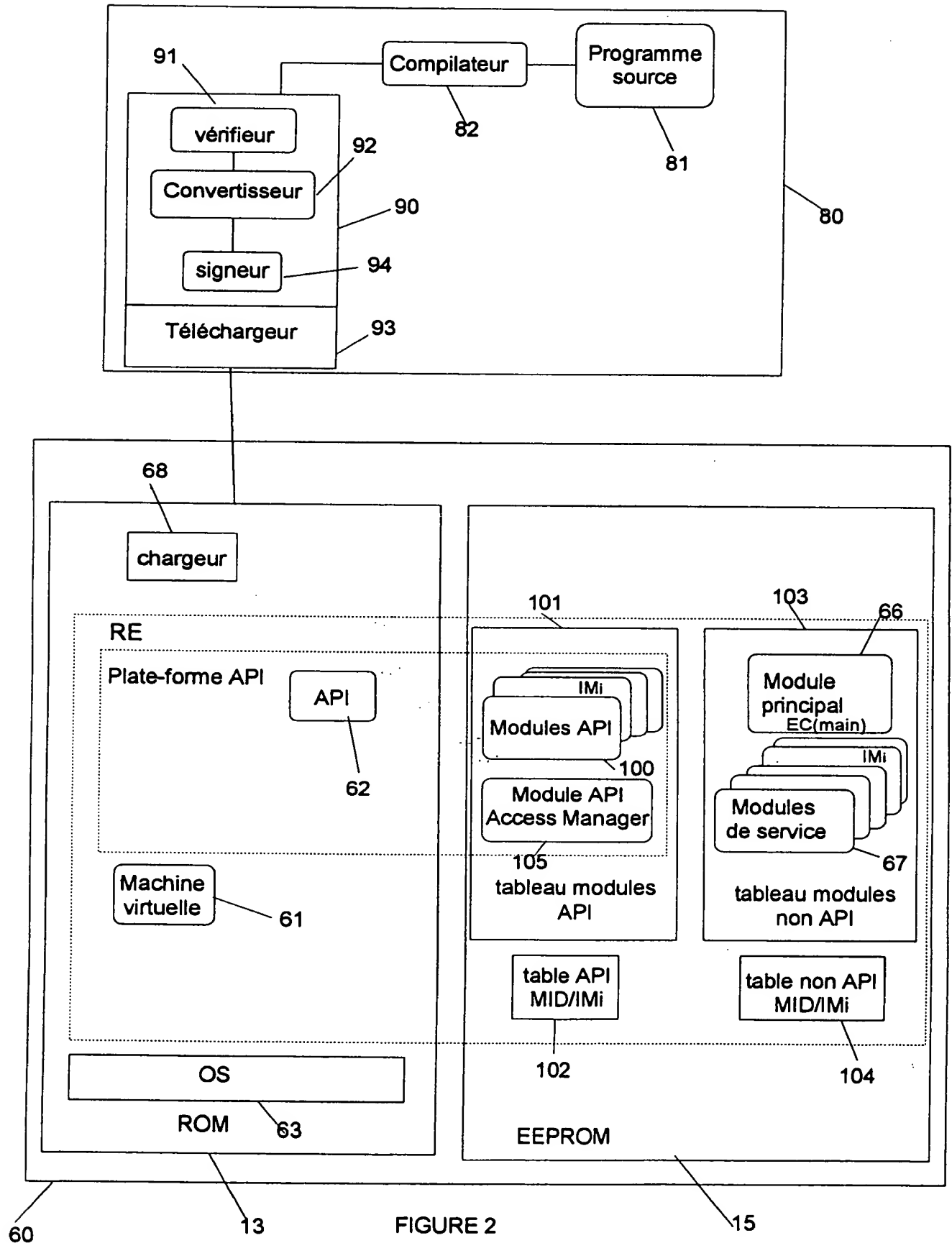
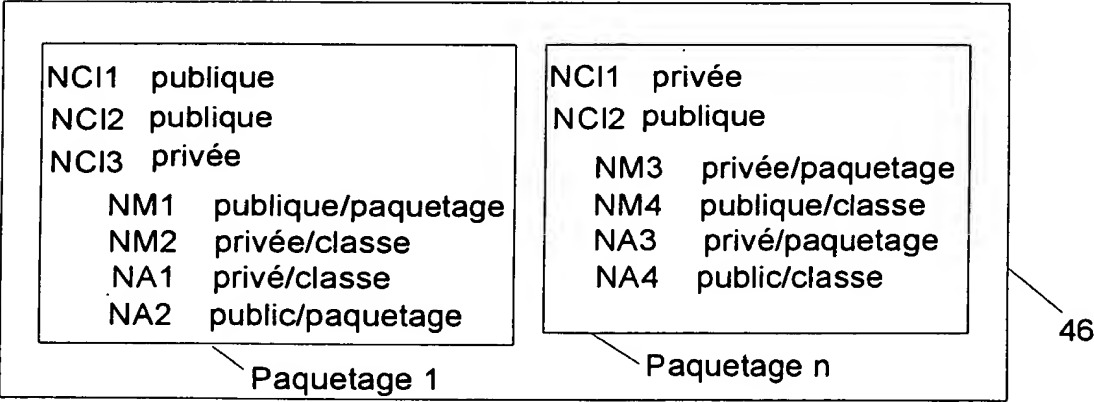
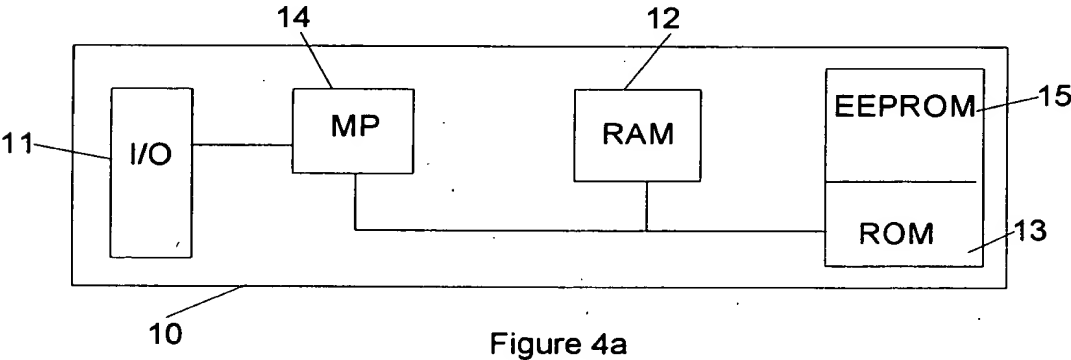
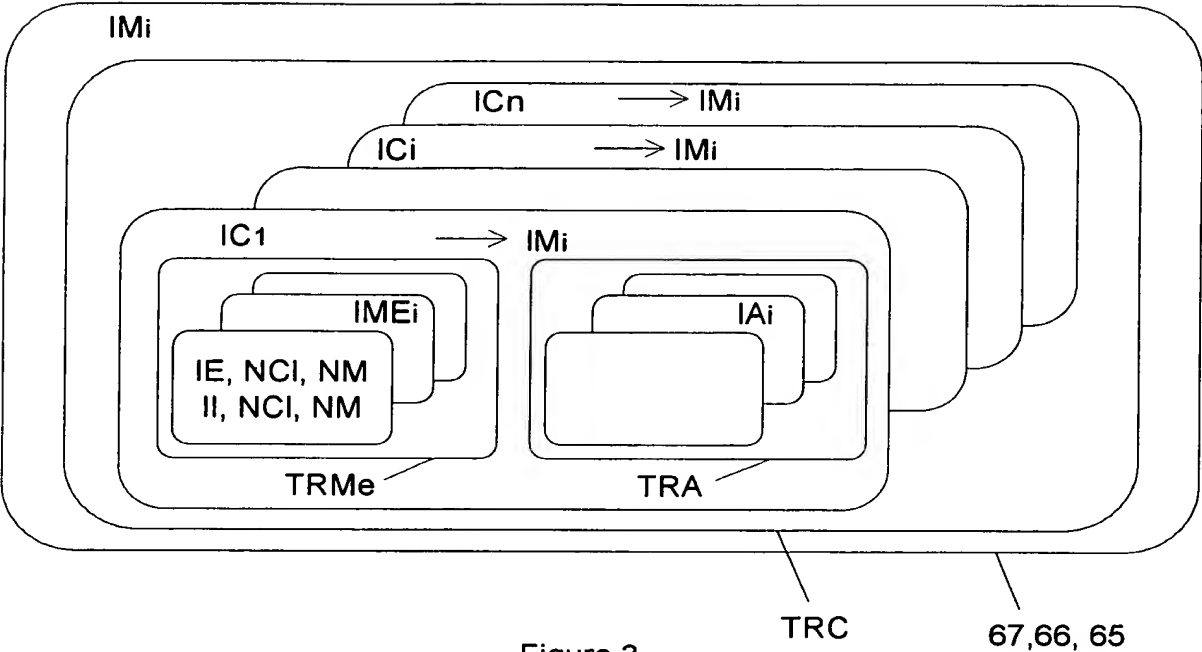


FIGURE 2

PL 3/4



## PL 4/4

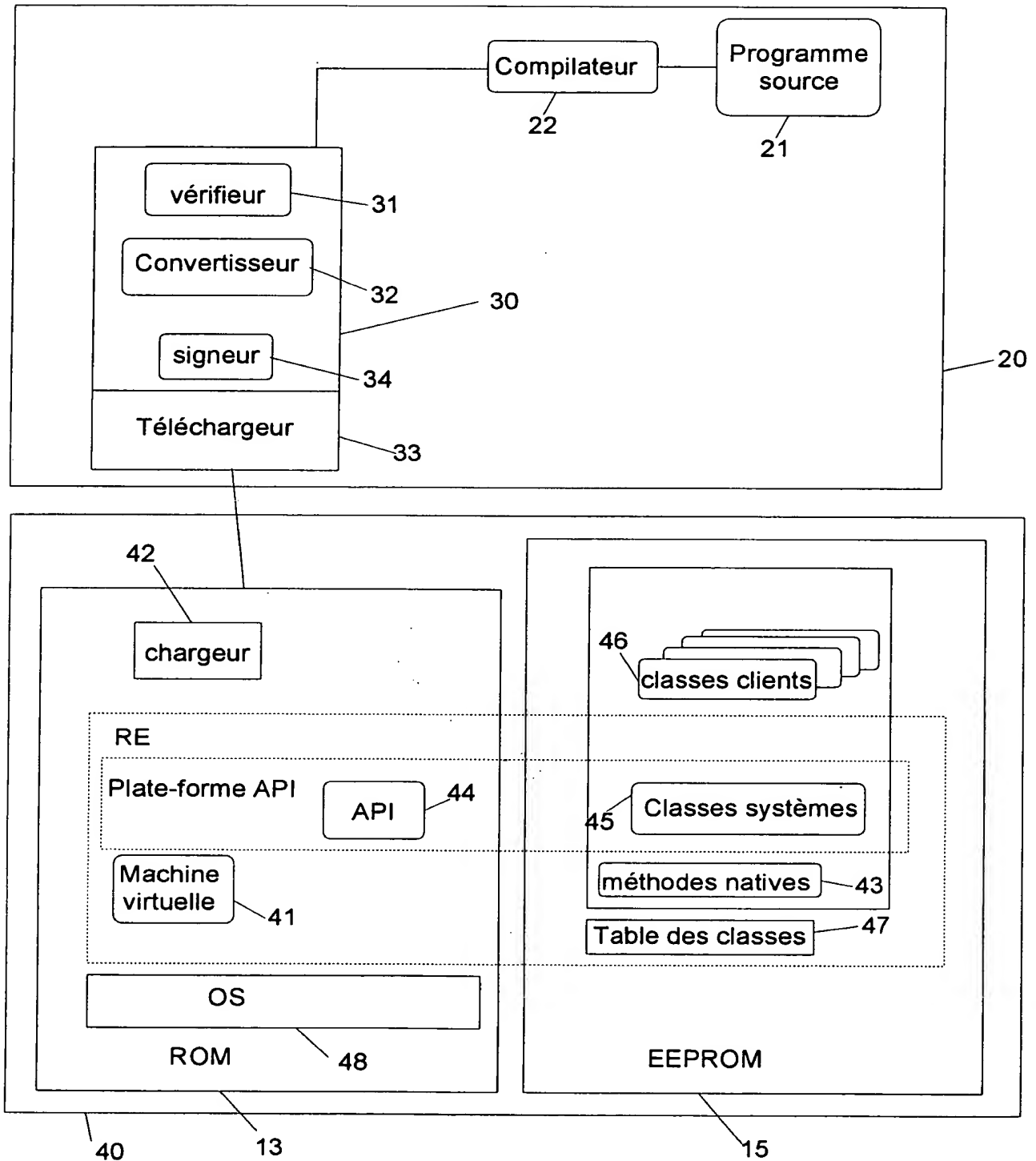


FIGURE 4b